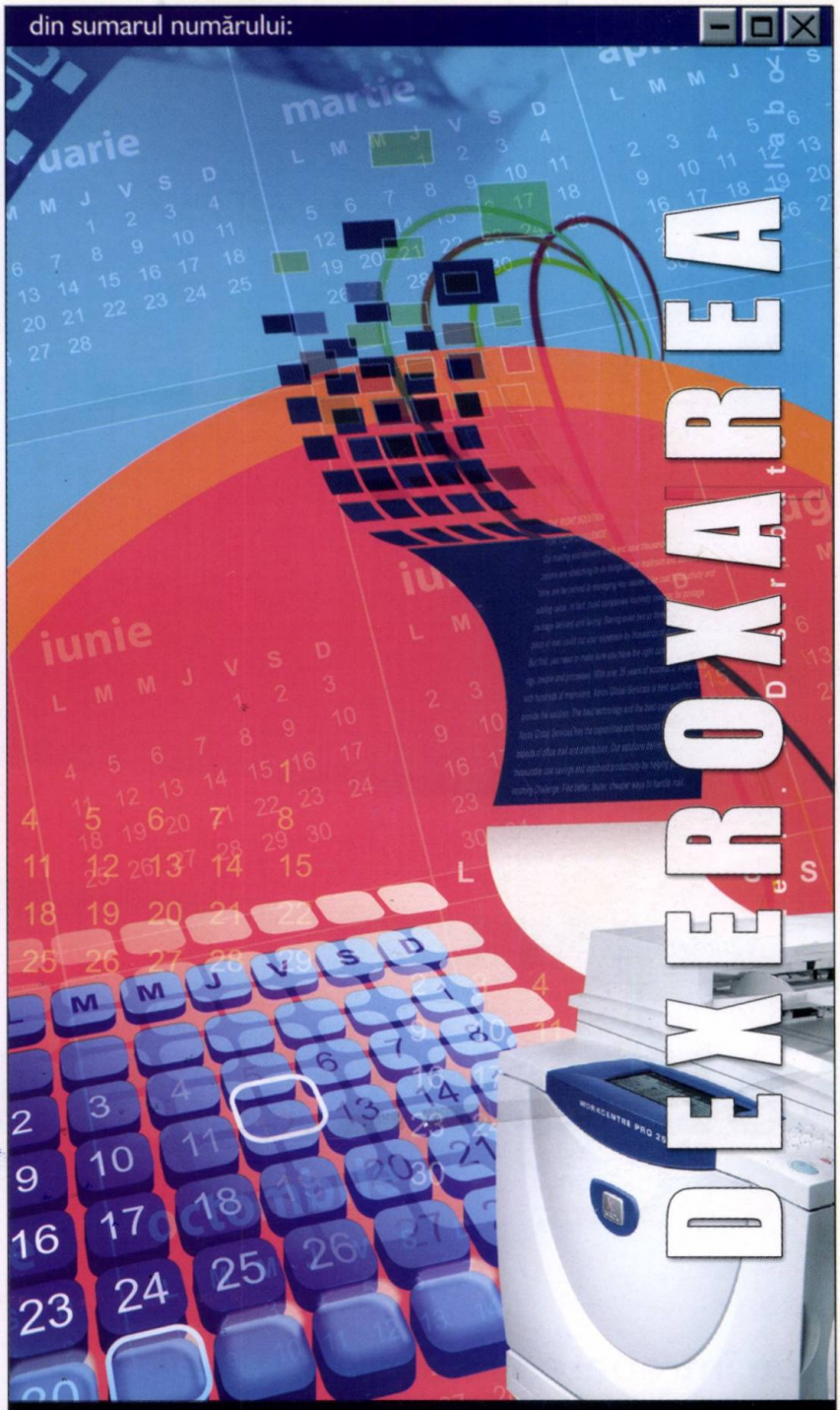
gfo
Ianuarie 2007
Vol. 17/1
40.000 LEI / 4 LEI NOI

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

O publicație Agora Media

din sumarul numărului:



probleme
Concurs internațional de programare

focus
Metoda BACKTRACKING

babel
Combinatorică în EXCEL
Problema CALENDARULUI

tehno
Deprotejarea fișierelor NOVELL DOS

5 948373 000645 01

THE EUROPEAN IST PRIZE
World Summit Award
Europe Awards

AeL. Simplu şi la obiect.

Lecţii interactive pentru noua generaţie.

Orice întrebare are un răspuns!

Noutăţi, suport tehnic, competiţii,
forum : www.portal.edu.ro

Orice problemă are o soluţie!

Centrul de suport tehnic de luni până
vineri între orele 8.00-20.00
Telverde : 0800410444 (linie gratuită)

Orice cauţi poate fi găsit!

Ai nevoie de o anumită lecţie electronică?
Întâi de toate, caut-o în biblioteca virtuală
AeL. Odată găsită, descarc-o gratuit!

AeL
Built to teach intelli-gently

sei

SIVECO
ROMANIA SA

Vol. 17, Nr. 1 /2007

gazeta de informatică

Sumar

actual.. PAG. 5

XEROX inventează DEXEROXAREA
Mihai SCORȚARU

probleme.. PAG. 6

Concursul INTERNAȚIONAL de PROGRAMARE ACM 2006/2007

Concursul INTERNAȚIONAL de PROGRAMARE NEMES TIHAMÉR

focus.. PAG. 16

Problema SUBȘIRULUI CRESCĂTOR de LUNGIME MAXIMĂ și APLICAȚII ale sale
Dan PRACSIU

PARTIȚIILE de LUNGIME FIXĂ ale unui NUMĂR natural
Tiberiu SOCACIU

Problema STEINHAUS-SIERPINSKI
Tiberiu SOCACIU

Demonstrarea CORECTITUDINII ALGORITMILOR care folosesc METODA BACKTRACKING
Tiberiu SOCACIU

babel.. PAG. 28

COMBINATORICĂ în EXCEL
László ZSAKÓ

Problema CALENDARULUI
Tiberiu SOCACIU

e-learning.. PAG. 35

Școala VIITORULUI

serial.. PAG. 36

VLAN, TRUNK și STP
Tiberiu SOCACIU

tehno.. PAG. 42

DEPROTEJAREA fișierelor sub NOVELL DOS
Tiberiu SOCACIU

begin.. PAG. 45

Principiul CONTORULUI
Tiberiu SOCACIU

MODALITATEA de REPARTIZARE COMPUTERIZATĂ a elevilor în LICEE pe baza unei MEDII calculate
Tiberiu SOCACIU

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tiberiu Socaciu
Colaboratori:
Pătcaș Csaba
Dan Pracsiu
László Zsakó

Adresa pentru corespondență:
Computer Libris Agora,
Șerpuitoare 25, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/534.911, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359
Editor: Agora Media SRL

Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NET Report*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

O publicație Agora MEDIA

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu

Difuzare:
Laurențiu Nicolae
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Cea mai bună introducere din lume în rețelele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine

EDIȚIA A PATRA

Rețele de calculatoare

ANDREW S. TANENBAUM

Cea mai bună introducere în arhitectura calculatoarelor - complet actualizată, reflectă ultimele schimbări din dinamica lumii a calculatoarelor

Andrew S. Tanenbaum

ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR

Ediția a IV-a

SISTEME DE OPERARE MODERNE

EDIȚIA A DOUA

Andrew S. Tanenbaum

Această carte de referință, bazată pe exemple concrete, oferă informații practice de primă mână pentru înțelegerea și construirea sistemelor de operare moderne.

THOMAS H. CORMEN CHARLES E. LEISERSON RONALD R. RIVEST

INTRODUCERE ÎN ALGORITMI

Cartea tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, fiind binevenita elevilor, studenților și profesorilor

CĂRȚILE ESENȚIALE ▸

...cărțile care îți însoțesc cariera

Comandă-le pe toate și le iei cu 20% mai ieftin.
Iar dacă ești student, cu 30%.

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

EURO și DOLARI

Leul a fost declarată cea mai puternică monedă din lume în anul 2006. Evoluția spectaculoasă a acestuia are repercusiuni destul de importante asupra vieții noastre de zi cu zi. Pentru majoritatea românilor creșterea valorii leului este un fapt pozitiv, pentru alții lucrurile nu sunt la fel de roz.

În acest număr nivelurile de dificultate ale articolelor sunt ilustrate de imagini ale elfilor întunecați (numiți și elfi ai nopții). Cum leul se apreciază peste noapte, poate acest fapt are o oarecare legătură cu subiectul acestui editorial.

În numărul următor, vom folosi imagini ale vrăjitorilor și cum leul are o apreciere magică, poate și acest fapt are o anumită legătură cu subiectul.

Sau poate că nu! Cert este că dolarul american are acuma aceeași valoare pe care o avea acum șase ani.

Așadar, în condițiile în care salariul mediu al românilor s-a cam dublat în ultimii șase ani, iar prețurile calculatoarelor sunt, în principiu, tot în dolari, ne este de cel puțin două ori mai ușor să ne achiziționăm acum un nou calculator.

De asemenea, factura la telefonul mobil este și ea mai mică (în termeni reali), o mașină costă și ea mai puțin etc.

Pentru mulți aceste lucruri sunt îmbucurătoare, dar nu pentru toți. Ca de obicei, raiul unora este infernul altora. Dacă importatorii sunt fericiți fiindcă plătesc mai puțin, exportatorii sunt nemulțumiți deoarece încasează mai puțin.

Dar aceste lucruri au fost discutate de cei în măsură să exprime păreri avizate, precum și de mulți care nu pricep nimic din ce se întâmălă. Eu voi încerca să mă refer la efectele pe care le are aprecierea leului asupra informaticienilor.

Să presupunem că, în 2001, un tânăr informatician se angajează la o companie *IT* de succes. În principiu, așa cum era atunci la modă, și cum este și acum, și-a negociat salariul în dolari.

Acum, după șase ani, are, cu siguranță, un salariu mai mare fiindcă, presupunem că o companie *IT* de succes își apreciază angajații și îi recompensează pe măsura meritelor. Așadar, dacă timp de șase ani nu l-au dat afară, înseamnă că sunt mulțumiți de el.

Dar, în ultimii șase ani, inflația cumulată a fost de peste 100%. Așadar, pentru a compensa pierderea datorată inflației, ar fi trebuit ca salariul să se dubleze în șase ani.

Nu spun că este imposibil, este chiar probabil, dar nu este cu siguranță sigur. Cu certitudine, o mulțime de programatori câștigă acum, în termeni reali, mai puțin decât au câștigat acum șase ani.

Dacă ne gândim la cei care s-au angajat acum doi-trei ani, situația este mai neplăcută. Dolarul valora mai mult de 3.5 lei noi, iar euro trecea de 4 lei. Acum, dolarul valorează aproximativ 2.6 lei, iar euro valorează cam 3.4 lei.

În funcție de moneda aleasă (pe atunci deja se obișnuia să se folosească euro), a pierdut 15 - 25%.

În această perioadă am avut și inflație, de 15 - 20%. Mă îndoiesc că majoritatea salariilor programatorilor au crescut cu 30 - 40% în ultimii doi-trei ani.

Evident, pe parcurs, unii erau fericiți fiindcă dolarul și euro creștea, deci salariile creșteau și ele.

Acum, salariile scad văzând cu ochii și mulți se declară nemulțumiți, fapt care nu este prea corect.

Este dificil de ajuns la un compromis; unele companii susțin, pe bună dreptate, că veniturile lor provin, în mare parte, din exporturi, deci este normal ca salariile să fie exprimate în euro sau dolari; aceste companii nu își permit să ajusteze salariile. Alte companii au fost nevoite să indexeze salariile pentru a reduce așa-zisele pierderi.

Nu pot să spun dacă nemulțumirile sunt sau nu justificate. Cu siguranță, este neplăcut ca veniturile să ți se reducă datorită variației cursului, mai ales că erai obișnuit ca ele să crească datorită acestei variații.

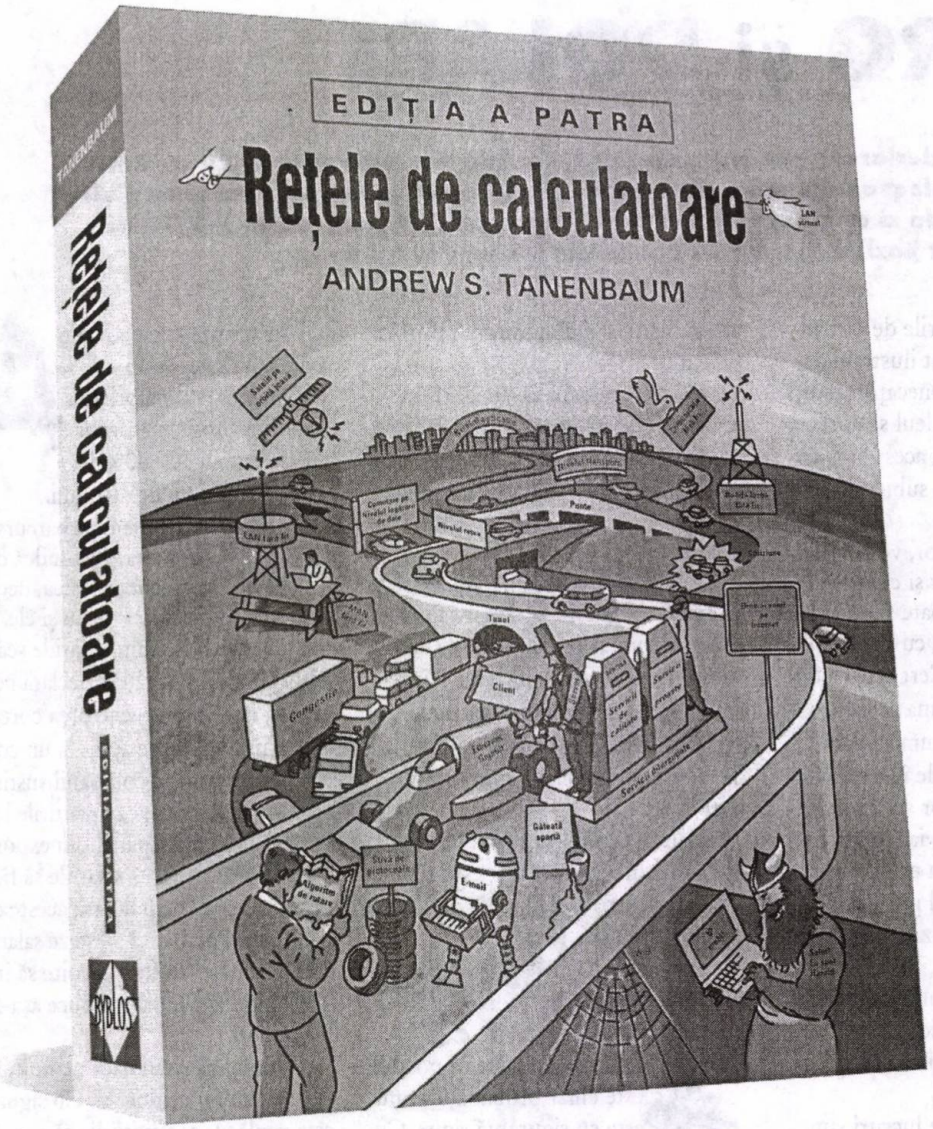
La urma urmei, este o loterie. Dacă ai acceptat un salariu în valută, ți-ai asumat și un risc valutar. Situația este alta dacă ți s-a impus salariul în valută.

Așadar, când vă veți negocia salariile de programatori, va trebui să aveți în vedere faptul că nu se știe cum vor evolua diversele valute. Puteți risca, sau puteți "miza pe leu", așa cum recomanda *BNR*.

Mihai Scorțaru

Cine are carte, are parte!

...de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețelele de calculatoare

- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o REDUCERE semnificativă!

BYBLOS

Byblos srl, tel: 0318-056-314, 0721-927-499, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librărie: Galeria NOI-La Calu' Bălan, Str. Halelor Nr. 3 (după Hanul lui Manuc).

XEROX inventează DEXEROXAREA

Mihai Scorțaru

Compania Xerox dezvoltă o tehnologie de listare care creează imagini care rezistă doar o zi. Apoi, hârtia poate fi refolosită, reducându-se astfel cantitatea de hârtie risipită în birouri.

Datorită faptului că multe documente sunt listate doar pentru a fi citite mai comod, ele sunt reciclate sau distruse foarte repede. Pentru a evita o astfel de risipă, *Xerox* a inventat o tehnologie de listare care creează imagini care rezistă o singură zi.

Tehnologie, denumită *Erasable Paper*, ar putea duce la reducerea semnificativă a cantității de hârtie utilizate în birouri, deoarece imediat ce imaginea dispare, hârtia poate fi utilizată din nou.

Xerox estimează că două pagini din cinci sunt utilizate doar în ziua în care au fost listate. Acestea conțin, de obicei, simple *e-mail*-uri sau alte materiale care sunt citite o singură dată.

Compania a declarat că a început procedurile de înregistrare a patentelor pentru această tehnologie.

Paul Smith

manager Xerox

În ciuda faptului că ne bazăm tot mai mult pe calculatoare pentru a procesa informații, depindem încă foarte mult de hârtie pentru a citi și a acumula cunoștințe.
Bineînțeles, toți am dori să folosim cât mai puțină hârtie dar, din discuțiile cu clienții noștri, știm că mulți preferă încă să lucreze cu informații imprimate pe hârtie. Documentele care se șterg singure și care pot fi folosite doar pentru scurt timp, reprezintă cel mai bun compromis între cele două lumi.

Eric Shrader

PARC area manager

Acesta va rămâne un proiect de cercetare pentru un timp.
Experimentele noastre demonstrează că această tehnologie este viabilă și am făcut un prim pas, dar acesta este nu este singurul pas care trebuie făcut pentru a dezvolta un sistem care va fi disponibil publicului larg.

Această tehnologie experimentală de listare a fost rezultatul unei colaborări dintre *Xerox Research Centre* din *Canada* și *Palo Alto Research Center*.

Potrivit unui comunicat de presă, momentul decisiv al cercetării s-a petrecut atunci când au fost descoperiți compuși care își schimbă culoarea atunci când absorb lumină cu o anumită lungime de undă, dar noua culoare dispare treptat.

În acest moment, hârtia se șterge după un interval de timp cuprins între 16 și 24 de ore; după ștergere hârtia poate fi refolosită.

Cercetătorii de la *Xerox Research Centre* lucrează la cerneala care va fi utilizată, în timp ce cei de la *PARC* lucrează la crearea unui aparat care poate lista pe hârtia specială.

Se lucrează la un prototip pentru o imprimantă care utilizează o bară de lumină (cu o lungime de undă specială) pentru a crea imaginea.

București, octombrie 2006

Concursul INTERNAȚIONAL de programare ACM 2006/2007

Etapa regională pentru Europa de Sud-Est a Concursului de Programare ACM s-a desfășurat pe data 21 octombrie 2006, la București. Vă prezentăm enunțurile celor nouă probleme cu care s-au confruntat reprezentanții celor 43 de echipe de la 31 de universități din șapte țări.

P070101: Sudoku

Intrare: fișierul `A.IN`
Ieșire: ieșirea standard
Denumirea sursei: `A.C`, `A.CPP` sau `A.JAVA`

O grilă *Sudoku* este un caroiaj de dimensiuni 16 × 16, format din celule grupate în șaisprezece pătrate de dimensiune 4 × 4; anumite celule conțin litere cuprinse între A și P (primele 16 litere mari ale alfabetului englezesc), așa cum se arată în figura 1.

		A					C						O		I
	J			A		B		P		C	G	F		H	
		D			F		I		E					P	
	G		E	L		H					M		J		
				E					C			G			
	I			K		G	A		B				E		J
D		G	P			J		F					A		
	E				C		B			D	P			O	
E			F		M			D			L		K		A
	C									O		I		L	
H		P		C			F		A			B			
			G		O	D				J					H
K				J					H		A		P		L
		B			P			E			K			A	
	H			B			K			F	I		C		
		F				C			D			H		N	

Figura 1: O grilă Sudoku

Jocul constă în completarea celulelor cu litere de la A la P, astfel încât fiecare literă să apară o singură dată pe o linie, o coloană sau în pătratul de dimensiuni 4 × 4 în care se află. Conținutul inițial al grilei satisface restricțiile amintite anterior și garantează existența unei soluții unice.

Soluția corespunzătoare grilei din figura 1 este prezentată în figura 2.

Cerință

Scrieți un program pentru acest joc.

		A					C						O		I
	J			A		B		P		C	G	F		H	
		D			F		I		E					P	
	G		E	L		H					M		J		
				E					C			G			
	I			K		G	A		B				E		J
D		G	P			J		F					A		
	E				C		B			D	P			O	
E			F		M			D			L		K		A
	C									O		I		L	
H		P		C			F		A			B			
			G		O	D				J					H
K				J					H		A		P		L
		B			P			E			K			A	
	H			B			K			F	I		C		
		F				C			D			H		N	

Figura 2: Soluția grilei din figura 1

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `A.IN`.

Fiecare set de date descrie o grilă și conține 16 șiruri de caractere pe 16 linii consecutive ale fișierului, așa cum se arată în exemplul de mai jos.

Cel de-al *i*-lea șir de caractere reprezintă cea de-a *i*-a linie a grilei.

Fiecare șir conține câte 16 caractere și începe pe prima poziție a liniei.

Caracterele care formează șirul fac parte din mulțimea {A, B, C, ..., P, -}, unde caracterul - (minus) indică o celulă goală.

Seturile de date sunt separate printr-o singură linie vidă și se încheie odată cu marcajul de sfârșit de fișier.

Date de ieșire

Programul va tipări soluția la ieșirea standard folosind același format ca în cazul datelor de intrare.

Exemplu

`A.IN`

```
--A----C-----O-I
-J--A-B-P-CGF-H-
--D--F-I-E----P-
-G-EL-H----M-J--
----E----C--G---
-I--K-GA-B---E-J
D-GP--J-F----A--
-E---C-B--DP--O-
E--F-M--D--L-K-A
-C--------O-I-L-
H-P-C--F-A--B---
---G-OD---J----H
K---J----H-A-P-L
--B--P--E--K--A-
-H--B--K--FI-C--
--F---C--D--H-N-
```

Ieșire

```
FPAHMJECNLBDKOGI
OJMIANBDPKCGFLHE
LNDKGFOIJEAHMBPC
BGCELKHPOFIMAJDN
MFHBELPOACKJGNID
CILNKDGAHBMOPEFJ
DOGPIHJMFNLECAKB
JEKAFCNBGIDPLHOM
EBOFPMIJDGHLNKCA
NCJDHBAEKMOFIGLP
HMPLCGKFIAENBDJO
AKIGNODLBPJCEFMH
KDEMJIFNCHGAOPBL
GLBCDPMHEONKJIAF
PHNOBALKMJFIDCEG
IAFJOECGLDPBHMNK
```

P070102: Subsecvență

Intrare: fișierul `B.IN`
Ieșire: ieșirea standard
Denumirea sursei: `B.C`, `B.CPP` sau `B.JAVA`

Se consideră o secvență formată din N numere întregi pozitive ($10 < N < 100.000$), fiecare dintre ele fiind cel mult egal cu 10000, precum și un număr întreg pozitiv S ($S < 100.000.000$).

Cerință

Scrieți un program care determină lungimea minimă a unei subsecvențe de elemente consecutive a căror sumă este mai mare sau egală cu S.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `B.IN`.

Pentru fiecare set de date se vor citi de pe prima linie valorile N și S, care sunt separate printr-un spațiu.

Cele N numere sunt date pe cea de-a doua linie corespunzătoare setului de date și sunt separate prin spații.

Seturile de date se încheie odată cu marcajul de sfârșit de fișier.

Date de ieșite

Pentru fiecare set de date, programul dumneavoastră va trebui să scrie la ieșirea standard, pe câte o linie, răspunsul corect.

Exemplu

`B.IN`

```
10 15
5 1 3 5 10 7 4 9 2 8
5 11
1 2 3 4 5
```

Ieșire

```
2
5
```

P070103: "Cine știe câștigă" cu vedete

Intrare: fișierul `C.IN`
Ieșire: ieșirea standard
Denumirea sursei: `C.C`, `C.CPP` sau `C.JAVA`

Este destul de greu să inventezi o problemă care este suficient de simplă pentru ca oricine să o poată rezolva, dar suficient de dificilă pentru a fi demnă de un anumit respect (minim).

De obicei greșim de o parte au de alta. Cât de simplă poate fi o problemă?

La concursul *Cine știe câștigă* pentru vedete întrebările și răspunsurile sunt oarecum confuze și, datorită faptului că participanții sunt vedete, este cu adevărat necesar ca întrebările să fie simple.

Cerință

Scrieți un program care determină întrebările (ecuații care trebuie rezolvate) pe baza răspunsurilor.

Mai exact, programul va trebui să determine cea mai simplă ecuație care are ca rezultat răspunsul dat, dintre toate ecuațiile care folosesc simboluri matematice standard în maniera obișnuită.

În contextul de față, noțiunea "cea mai simplă" poate fi definită în mod neambiguu în mai multe moduri, toate ducând la același rezultat.

Pentru moment, va trebui să determinați acea ecuație a cărei transformare în răspunsul dorit de organizatori necesită efort minim.

De exemplu, pentru răspunsul `X = 2`, putem crea ecuația `9 - X = 7`. Alternativ, am putea crea sistemul `X > 0; X ^ 2 = 4`.

Acestea ar putea să nu fie cele mai simple exuații posibile. Rezolvarea unor probleme ar putea fi dificilă pentru vedete.

Date de intrare

Fișierul de intrare **`C.IN`** conține mai multe linii, pe fiecare dintre ele aflându-se o soluție sub forma *<simbol> = <valoare>*.

Date de ieșire

Pentru fiecare linie din fișierul de intrare se va scrie la ieșirea standard cel mai simplu sistem de ecuații care duce la obținerea soluției date, respectând modul de folosire al spațiilor exact ca la intrare.

Exemplu

`C.IN`

```
Y = 3
X=9
```

Ieșire

```
Y = 3
X=9
```

P070104: Sherlock Holmes

Intrare: fișierul **`D.IN`**
Ieșire: ieșirea standard
Denumirea sursei: **`D.C`**, **`D.CPP`** sau **`D.JAVA`**

Renumitul detectiv *Sherlock Holmes* trebuie să găsească o ieșire dintr-o situație complicată.

El are n cutii, C_1, C_2, ..., C_n, unde n este un număr par, și fiecare cutie conține un număr total de m bile, care pot fi albe sau negre.

Notăm prin $C_i = (A_i, N_i)$ o cutie care conține A_i bile albe și N_i bile negre.

Sherlock Holmes trebuie să împartă cutiile în două mulțimi, fiecare fiind formată din $n / 2$ cutii, astfel încât, în fiecare dintre cele două mulțimi formate, fie bilele albe, fie bilele negre, dețin majoritatea.

Cerință

Notăm prin m_1 și m_2 procentajul bilelor majoritare din fiecare dintre cele două mulțimi.

Holmes trebuie să determine cât mai repede maximul posibil al valorii $\min(m_1, m_2)$. Îl puteți ajuta?

Date de intrare

Fișierul de intrare **`D.IN`** conține mai multe seturi de date.

Primul număr corespunzător unui set de date este n ($n < 10000$) și indică numărul total al cutiilor.

Urmează numărul m ($m < 10000$) al bilelor din fiecare cutie, iar apoi, pentru fiecare cutie, numărul bilelor albe și cel al bilelor negre din fiecare cutie.

Spațiile pot apărea liber în cadrul datelor de intrare. Datele de intrare sunt corecte și se încheie odată cu marcajul de sfârșit de fișier.

Date de ieșire

Pentru fiecare linie din fișierul de intrare se va scrie la ieșirea standard culoarea bilelor majoritare (`W` pentru bilele albe și `B` pentru cele negre), urmată de valoarea maximă cerută.

În cazul în care majoritatea cerută nu există, pe linia corespunzătoare se va scrie mesajul `No solution`.

Exemplu

`D.IN`

```
4
30
17 13
12 18
20 10
14 16
```

Ieșire

```
W 51.67
```

Explicație

În cadrul exemplului anterior avem un singur set de date. Există patru cutii, care conțin fiecare un număr de 30 de bile.

De exemplu, prima cutie conține 17 bile albe și 13 bile negre.

Există o singură partiționare posibilă a cutiilor și anume (C_1, C_4) formează o mulțime, iar (C_2, C_3), o formează pe cealaltă.

Bilele albe dețin majoritatea în fiecare dintre cele două mulțimi determinate.

P070105: Sistem de plată

Intrare: fișierul **`E.IN`**
Ieșire: ieșirea standard
Denumirea sursei: **`E.C`**, **`E.CPP`** sau **`E.JAVA`**

Sistemul de plăți de la *Universitatea pentru Producerea Apei Minerale* este complet automat (scrie integrat în lim-

bajul de programare *Tomato*) și permite introducerea sumei care se dorește a fi retrasă.

Datorită faptului că salariile sunt foarte mari, profesorii pot introduce suma într-o formă exponențială. Așadar, dacă dorim să retragem 16 *UAM* (unități de apă minerală) putem introduce `16`, `2^4` sau `2^2^2`.

Într-o zi *Stănescu* a dorit să retragă o sumă de bani; el avea în cont 80 *UAM*.

Din greșeală, el a introdus suma `2^3^2` și, spre surprinderea sa, a primit 512 *UAM*, deși nu ar fi trebit să poată retrage decât cel mult 80.

Sistemul este format din două mari module - primul modul verifică dacă există fonduri suficiente în cont pentru ca tranzacția să poată fi efectuată, iar cel de-al doilea modul livrează banii.

Se pare că primul modul are o problemă cu operatorul `^`. Acest operator este evaluat de la stânga la dreapta, în timp ce al doilea modul îl evaluează (corect) de la dreapta la stânga.

Așadar, pentru primul modul valoarea introdusă este `2^3^2=(2^3)^2=64`; pentru al doilea modul, valoarea este considerată a fi `2^3^2=2^(3^2)=512`.

Cerință

Va trebui să scrieți un program care să îi permită lui *Stănescu* să profite cât mai mult de această situație și să retragă cât mai mulți bani din sistemul de plăți.

Dacă credeți că o astfel de procedură este ilegală (sau imorală), trebui să știți că *Universitatea pentru Producerea Apei Minerale* este o instituție malefică, plină de persoane corupte.

Date de intrare

Fișierul de intrare **`E.IN`** conține mai multe seturi de date.

Fiecare set de date constă într-un singur număr, scris pe o linie separată, care reprezintă suma din contului lui *Stănescu* (cuprinsă între 1 și 10^{100} - 1).

Date de ieșire

Pentru fiecare linie din fișierul de intrare se va scrie la ieșirea standard comanda care trebuie introdusă de *Stănescu* pentru ca suma obținută să fie maximă.

Trebuie respectate următoarele condiții:

- comanda constă doar din numere întregi, între care se află operatorul `^`;
- comanda trece de verificarea efectuată de primul modul și determină cel de-al doilea volum să furnizeze o sumă cât mai mare;
- comanda nu conține numărul `1` (care ar fi oricum nefolositor).

Dacă există mai multe răspunsuri posibile, trebuie ales cel pentru care primul număr este minim.

Dacă, în continuare, există mai multe răspunsuri posibile, trebuie ales cel pentru care al doilea număr este minim și așa mai departe.

Exemplu

`E.IN`

```
16
80
49
1025
12341234
12345678901234567890
```

Ieșire

```
2^2^2
2^3^2
7^2
2^2^5
2^2^2^5
2^2^2^2^3^2
```

P070106: Porți stelare

Intrare: fișierul **`F.IN`**
Ieșire: ieșirea standard
Denumirea sursei: **`F.C`**, **`F.CPP`** sau **`F.JAVA`**

"Cu mult timp în urmă, într-o galaxie foarte, foarte îndepărtată" o civilizație avansată a descoperit o modalitate prin care se putea călători instantaneu între diferite sisteme stelare. Din acel moment, ei și-au dedicat toate eforturile pentru a construi perechi de porți stelare care leagă planete îndepărtate.

În scurt timp, rețeaua de transport a devnit atât de complexă încât au nevoie de ajutor pentru a păstra informații referitoare la lumile legate prin porți stelare.

Cerință

Va trebui să scrieți un program care va ajuta la păstrarea informațiilor referitoare la sistemele stelare conectate. Noi ne vom asigura că informațiile determinate de program vor ajunge în continuumul spațio-temporal al civilizației.

Două planete A și B sunt conectate dacă există o poartă stelară directă care le leagă sau dacă există o secvență de planate $P_1, P_2, ..., P_n$, unde $P_1 = A$, $P_n = B$ și există o poartă stelară directă între oricare două planete P_k și P_{k-1}, unde $k \in \{2, 3, ..., n\}$.

Conexiunile sunt bidirecționale. Pot exista mai multe drumuri prin care sunt conectate două planete.

Date de intrare

Fișierul de intrare **`F.IN`** conține mai multe seturi de date.

Fiecare set de date ocupă una sau mai multe linii. Fișierul de intrare nu conține linii vide.

Fiecare linie începe cu o literă care poate fi `D`, `C` sau `Q` (literă mare sau mică), literă care este urmată de unu până la cinci numere întregi, cu următoarele semnificații:

- `D` (definiție) are un singur argument care indică numărul N al planetelor pentru setul de date care urmează; sunt cel mult 6.000.000 de planete, acestea fiind numerotate de la 1 la N;

- `C` (conexiune): indică o conexiune între perechea sau perechile de planete date ca argumente;
- `Q` (interogare): cere verificarea faptului dacă perechea sau perechile de planete date ca argumente sunt sau nu conectate.

Sintaxa comenzilor `C` și `Q` (`X` în cele ce urmează) este aceeași:

- `X src dst`: descrie o conexiune sau o interogare, corespunzătoare perechii date de planete (`src`, `dst`);
- `X src dst nnn`: descrie conexiuni sau interogări, corespunzătoare a `nnn` perechi de planete; prima planetă care formează aceste perechi este `src`, iar cea de-a doua planetă este `dst` pentru prima pereche și planetele cu următoarele numere de ordine, pentru următoarele perechi; de exemplu, `X 1 100 3` descrie conexiuni sau interogări pentru perechile (`1`, `100`), (`1`, `101`) și (`1`, `102`);
- `X src dst nnn step`: descrie conexiuni sau interogări, corespunzătoare a `nnn` perechi de planete; prima planetă care formează aceste perechi este `src`, iar cea de-a doua planetă este `dst` pentru prima pereche și planetele cu următoarele numere de ordine, folosind pasul `step`, pentru următoarele perechi; de exemplu, `X 1 100 3 5` descrie conexiuni sau interogări pentru perechile (`1`, `100`), (`1`, `105`) și (`1`, `110`);
- `X src dst nnn dststep srcstep`: descrie conexiuni sau interogări, corespunzătoare a `nnn` perechi de planete; prima planetă din prima pereche este `src`, iar pentru următoarele perechi prima pereche este dată de următoarele planete, cu pasul `srcstep`; cea de-a doua planetă este `dst` pentru prima pereche și planetele cu următoarele numere de ordine, folosind pasul `dststep`, pentru următoarele perechi; de exemplu, `X 1 100 3 5 15` indică conexiuni sau interogări pentru perechile (`1`, `100`), (`16`, `105`) și (`31`, `110`);

Date de ieșire

Pentru fiecare interogare din fișierul de intrare se va scrie la ieșirea standard câte o linie care va conține două numere separate prin trei caractere (un spațiu, o liniuță și un alt spațiu). Primul număr reprezintă numărul perechilor de planete conectate corespunzătoare interogării, iar cel de-al doilea reprezintă numărul perechilor de planete care nu sunt conectate.

Exemplu

`F.IN`

```
d 5
C 1 3
D 20
q 1 3
c 1 10 10
Q 1 2 18 1 1
```

Ieșire

```
0 - 1
9 - 9
```

P070107: Maxim

Intrare: fișierul **`G.IN`**
Ieșire: ieșirea standard
Denumirea sursei: **`G.C`**, **`G.CPP`** sau **`G.JAVA`**

Considerăm numerele reale $x_1, x_2, \ldots, x_m$ care satisfac, pentru numerele întregi a și b ($a > 0$) următoarele condiții:

- $-1 / a^{1/2} \le x_i \le a^{1/2}$;
- $x_1 + x_2 + \ldots + x_m = b \cdot a^{1/2}$.

Cerință

Determinați valoarea maximă a expresiei $x_1^p + x_2^p + \ldots + x_m^p$, unde p este un număr întreg par.

Date de intrare

Fișierul de intrare **`G.IN`** conține mai multe seturi de date.

Fiecare set de date ocupă o linie și constă în numerele întregi m, p, a și b ($m \le 2000$, $p \le 12$, p este număr par).

Datele de intrare sunt corecte; pentru fiecare set de date există cel puțin un șir $x_1, x_2, \ldots, x_m$ care satisface condițiile impuse.

Date de ieșire

Pentru fiecare linie din fișierul de intrare se va scrie la ieșirea standard câte un număr, care reprezintă valoarea maximă a expresiei.

Exemplu

`G.IN`

```
1997 12 3 -318
10 2 4 -1
```

Ieșire

```
189548
6
```

P070108: Japonia

Intrare: fișierul **`H.IN`**
Ieșire: ieșirea standard
Denumirea sursei: **`H.C`**, **`H.CPP`** sau **`H.JAVA`**

Japonia se pregătește să găzduiască finala *ACM* și, pentru acest evenimente, trebuie construite o mulțime de străzi.

Japonia este o insulă mare cu N orașe pe coasta de est și M orașe pe coasta de vest ($M, N \le 1000$).

Vor fi construite un număr total de K autostrăzi de mare viteză.

Orașele de pe fiecare coastă sunt numerotate, dinspre nord spre sud, începând de la 1.

Fiecare autostradă este reprezentată de o linie dreaptă care leagă un oraș de pe coasta de est de un oraș de pe coasta de vest.

Finațarea construcțiilor este garantată de *ACM*. O mare parte a sumei este determinată de numărul intersecțiilor. Cel mult două autostrăzi se pot intersecta într-un anumit punct.

Cerință
Scrieți un program care determină numărul total al intersecțiilor dintre autostrăzi.

Date de intrare
Fișierul de intrare `H.IN` conține, la început, numărul T al seturilor de date.

Fiecare set de date începe cu trei numere N, M și K. Pe fiecare dintre următoarele K linii se află câte două numere care identifică orașele legate de autostrada corespunzătoare. Primul dintre aceste numere identifică orașul de pe coasta de est, iar cel de-al doilea identifică orașul de pe coasta de vest.

Date de ieșire
Pentru fiecare set de date, la ieșirea standard se va scrie o linie care va avea următorul format:

`Test case` *<număr test>*`:` *<număr intersecții>*

Exemplu
`G.IN`

```
1
3 4 4
1 4
2 3
3 2
3 1
```

Ieșire

```
Test case 1: 5
```

P070109: Pereche de drumuri
Intrare: fișierul `I.IN`
Ieșire: ieșirea standard
Denumirea sursei: `I.C`, `I.CPP` sau `I.JAVA`

O companie de produse chimice se confruntă cu o problemă neobișnuită de drum minim. Există N depozite (vârfuri) în care sunt înmagazinate materialele. De asemenea, există M trasee de livrare (muchii) care leagă perechi de depozite. Fiecare traseu de livrarea are atașat un cost.

În cazul problemei obișnuite a drumului minim, compania ar fi dorit să efectueze un transport de la primul depozit (identificat prin 0) la ultimul depozit (identificat prin N - 1). Această problemă este ușor de rezolvat.

Problema companiei pare mai dificilă. Ei trebuie să efectueze două transporturi de la primul depozit la ultimul.

Produsele chimice sunt periculoase și nu pot fi transportate în siguranță cu același transport. Reglementările în vigoare obligă compania să folosească trasee diferite pentru cele două transporturi.

Mai mult, produsele transportate nu se pot afla în același depozit, decât folosind tehnologii speciale, disponibile doar în primul și în ultimul depozit.

Pentru început, compania ar dori să știe dacă este posibilă efectuarea transporturilor. Apoi, ei ar dori să determine costul total minim al celor două transporturi. Pe scurt, compania are nevoie de două trasee complet diferite de la primul la ultimul depozit, al căror cost total să fie minim.

Cerință
Scrieți un program care determină costul total minim sau determină faptul că transporturile nu pot fi efectuate.

Date de intrare
Fișierul de intrare `I.IN` conține mai multe seturi de date.

Prima linie corespunzătoare unui set de date conține numerele N ($N < 64$) și M ($M < 1000$), unde N reprezintă numărul depozitelor, iar M reprezintă numărul traseelor directe dintre două depozite.

Fiecare dintre următoarele M linii conține trei numere întregi i, j și v. O astfel de linie corespunde unui traseu direct; i și j reprezintă numerele de ordine ale depozitelor care delimitează traseul, iar v reprezintă costul efectuării unui transport de la depozitul i la depozitul j.

Aceste trasee sunt unidirecționale. Dacă un transport de la depozitul i la depozitul j are costul 10, acest fapt nu ne oferă nicio informație cu privire la costul unui transport de la depozitul j la depozitul i.

De asemenea, există posibilitatea existenței a două sau mai multe trasee diferite între oricare pereche de depozite.

O linie care conține două zerouri va marca sfârșitul datelor de intrare; acestă linie nu trebuie procesată.

Date de ieșire
Pentru fiecare set de date, la ieșirea standard se va scrie o linie care va avea următorul format:

`Instance #`*<număr test>*`:` *<cost total>*

În cazul în care transportul nu este posibil, în locul costului total se va scrie textul `Not possible`.

Exemplu
`G.IN`

```
2 1
0 1 20
2 3
0 1 20
0 1 20
1 0 10
4 6
0 1 22
1 3 11
0 2 14
2 3 26
0 3 43
0 3 58
0 0 0
```

Ieșire

```
Instance #1: Not possible
Instance #2: 40
Instance #3: 73
```

Noiembrie 2006

Concursul INTERNAȚIONAL de programare NEMES TIHAMÉR

Concursul "Nemes Tihamér" este organizat pentru elevi de liceu de către Comitetul Olimpic de Informatică din Ungaria. La acest concurs - prin tradiție - participă și elevi din România. În cele ce urmează redacția Ginfo pune la dispoziția celor interesați enunțurile problemelor primei etape.

Pe data de 9 noiembrie 2006 s-a desfășurat prima rundă a concursului de programare *Nemes Tihamér*.

În această fază elevii au avut de rezolvat probleme pe foaie și puteau să obțină cel mult 100 de puncte, o pătrime din punctele obținute urmând să fie transferate pentru runda următoare.

Concurenții au fost împărțiți în trei categorii: clasele V - VIII, clasele IX - X și clasele XI - XII.

La fel ca anul trecut, subiectele ultimelor două categorii s-au deosebit printr-o problemă propusă celor din a treia categorie, de către *Pătcaș Csaba*.

Categoria I

P070110: Formulă

(20 de puncte)

Următorul fragment de cod calculează valorile `a`, `b`, `c` și `d` pe baza valorii variabilelor `x` și `y`.

```
p := x + y;
q := x - y;
r := Abs(p);
s := Abs(q)
a := (p + s) / 2;
b := (p - s) / 2
c := Abs(s - r) / 2;
d := x - (q + s) / 2
```

A. Care va fi valoarea variabilelor `a`, `b`, `c` și `d`, dacă avem `x = 8` și `y = -5`?

B. Descrieți (în general) modul în care valorile `a`, `b`, `c` și `c` depind de `x` și `y`!

P070111: Roșii

(19 puncte)

N roșii se află una lângă cealaltă, de-a lungul unei linii. În timpul procesului de coacere, la un moment dat unele devin coapte. Roșiile se coc după următoarele reguli:

- O roșie necoaptă se coace într-o unitate de timp dacă, atât la dreapta cât și la stânga, cea mai apropiată roșie coaptă se află la aceeași distanță, dar nu este vecină.
- Dacă distanța dintre primele două roșii coapte este *K*, atunci roșia necoaptă, care se află la o distanță *K* - 1 de prima roșie coaptă la stânga, va deveni coaptă.
- Dacă distanța dintre ultimele două roșii coapte este *K*, atunci roșia necoaptă, care se află la o distanță *K* + 1 de ultima roșie coaptă, va deveni coaptă.

Pentru situațiile de mai jos, scrieți sub literele `v` (reprezentând roșiile necoapte) unitatea de timp când acestea devin coapte. Literele `c` reprezintă roșiile coapte.

A: vvvvcvvvvvvvccvvvcvvvvvv
0 00 0
B: vvvcvvvvvccvvvcvcvvvvvvv
0 00 0 0

Exemplu

Momentul 0: vvvvvcvvvc
Momentul 1: vvcvvcvcvc
Momentul 2: cvcvvcvcvc
Valorile: 2 1 0 1 0

P070112: În două sensuri

(24 de puncte)

Următorii doi algoritmi analizează literele cuvântului S, pornind simultan de la început spre sfârșit și de la sfârșit spre început.

```
algoritm Alfa(S)
  i ← 1
  j ← lungime(S)
  cât timp i ≤ j și S_i = S_j execută
    i ← i + 1
    j ← j - 1
  sfârșit cât timp
  dacă i > j atunci
    scrie 'OK'
  altfel
    scrie 'NOK'
  sfârșit dacă
sfârșit algoritm

algoritm Beta(S)
  i ← 1
  j ← lungime(S)
  cât timp i ≤ j și S_i ≠ S_j execută
    i ← i + 1
    j ← j - 1
  sfârșit cât timp
  dacă i ≤ j atunci
    scrie i
  altfel
    scrie 'NOK'
  sfârșit dacă
sfârșit algoritm
```

A. Ce afișează algoritmii de mai sus, dacă S = 'kertel'?

B. Pentru care cuvinte generează algoritmul Alfa rezultatul 'OK'? Dați un exemplu concret și formulați în general!

C. Pentru care cuvinte afișează algoritmul Beta un număr și ce reprezintă acest număr? Dați exemplu concret și formulați în general!

P070113: Doi indici

(34 de puncte)

Următorii trei algorimi încearcă să determine maximul (într-un anumit sens) elementelor șirului X, care conține N numere pozitive.

```
algoritm Primul(N, X, A, B)
  A ← 0
  B ← 0
  X_0 ← -1
  pentru i ← 1, N execută
    dacă X_i > X_A atunci
      A ← i
    sfârșit dacă
  sfârșit pentru
  pentru i ← 1, N execută
    dacă X_i > X_B și X_i < X_A atunci
      B ← I
    sfârșit dacă
  sfârșit pentru
sfârșit algoritm

algoritm AlDoilea(N, X, A, B)
  A ← 0
  B ← 0
  X_0 ← -1
  pentru i ← 1, N execută
    dacă X_i > X_A atunci
      B ← A
      A ← i
    sfârșit dacă
  sfârșit pentru
  pentru i ← 1, N execută
    dacă X_i > X_B atunci
      B ← I
    sfârșit dacă
  sfârșit pentru
sfârșit algoritm

algoritm AlTreilea(N, X, A, B)
  A ← 0
  B ← 0
  X_0 ← -1
  pentru i ← 1, N execută
    dacă X_i ≥ X_A atunci
      B ← A
      A ← i
    sfârșit dacă
  sfârșit pentru
sfârșit algoritm
```

A. Care va fi valoarea variabilelor A și B pentru fiecare algoritm, dacă N = 6 și X = (3, 5, 2, 5, 4, 5)?

B. Dați un șir X pentru N = 5, pentru care primul algoritm menține B = 0!

C. Dați un șir X pentru N = 5, pentru care al treilea algoritm menține B = 0!

D. Ce valori vor conține variabilele A și B după executarea fiecărui algoritm!

Categoriile a II-a și a III-a

P070114 : Circuit

(14 puncte pentru categoria a II-a, 10 puncte pentru categoria a III-a)

Un mesaj de patru biți este alcătuit din biții $b_1b_2b_3b_4$. La transmiterea mesajului poate să apară cel mult o eroare, adică cel mult un bit 1 se poate schimba într-un bit 0, sau un bit 0 se poate schimba într-un bit 1.

Pentru găsirea acestora, determinăm valorile altor trei biți în felul următor: $b5 = (b_2 + b_3 + b_4) \bmod 2$; $b_6 = (b_1 + b_3 + b_4) \bmod 2$; $b_7 = (b_1 + b_2 + b_4) \bmod 2$.

Astfel, codul transmis va fi: $b_1b_2b_3b_4b_5b_6b_7$.

A. Care dintre mesajele 1100011, 1010110, 1111111, 0101010 și 0110011 sunt corecte și care sunt greșite?

B. Dați codul pentru mesajele 1001 și 1010!

P070115: Analiza cifrelor

(23 de puncte pentru categoria a II-a, 18 puncte pentru categoria a III-a)

Următorii doi algoritmi determină valorile lui k, v și h pe baza vectorului X, format din N elemente cu valori cuprinse între 0 și 9.

```
algoritm Alfa(N, X, k, v, h)
  k ← 1
  v ← 1
  h ← 1
  j ← 1
  a ← 1
  pentru i ← 1, N execută
    dacă X_i = X_i+1 atunci
      a ← a + 1
    altfel
     dacă a > h atunci
         k ← j
         v ← i
         h ← a
     sfârșit dacă
        a ← 1
        j ← i + 1
    sfârșit dacă
   sfârșit pentru
sfârșit algoritm
```

```
algoritm Beta(N, X, k, h):
  T ← (0, ..., 0)
  k ← 0
  h ← 0
  pentru i ← 1, N execută
    T_X_i ← T_X_i + 1
    dacă T_X_i > h atunci
      k ← X_i
      h ← T_X_i
   sfârșit dacă
  sfârșit pentru
sfârșit algoritm
```

A. Care vor fi valoarile k, v și h după executarea algoritmilor de mai sus, dacă avem N = 10 și X = (1, 3, 5, 5, 5, 4, 3, 3, 2, 3)?

B. Dați enunțuri pentru problemele rezolvate de cei doi algoritmi; precizați în ce mod depind valorile k, v și h de datele de intrare!

P070116: Reuniune

(17 puncte pentru categoria a II-a, 13 puncte pentru categoria a III-a)

Mulțimea {1, 2, ..., N} se împarte în submulțimi disjuncte. Aceste submulțimi sunt descrise în vectorul T în felul următor: din fiecare submulțime considerăm un element care identifică submulțimea, adică fiecărui element i al submulțimii îi corespunde același T_i.

Dacă, de exemplu, avem submulțimea {3, 5} și elementul identificator este 5, atunci în vectorul T vom avea T_3 = 5 și T_5 = 5. Dacă o submulțime are un singur element, vom avea T_i = -1. Pentru reuniunea a două submulțimi este necesară doar cunoașterea celor doi identificatori:

```
algoritm Reuniune(A, B)
  dacă T_A = -1 atunci
    dacă T_B = -1 atunci
      T_A ← B
      T_A ← B
    altfel
      T_A ← T_B
    sfârșit dacă
  altfel
   dacă T_B = -1 atunci
     T_B ← T_A
   altfel
     X ← T_B
     pentru i ← 1, N execută
       dacă T_i = X atunci
         T_i ← T_A
    sfârșit dacă
   sfârșit pentru
 sfârșit dacă
sfârșit algoritm
```

A. Specificați conținutul vectorului T având 10 elemente după fiecare dintre următoarele operații: Reuniune(1, 5), Reuniune(2, 4), Reuniune(8, 9), Reuniune(1, 9), Reuniune(8, 5), Reuniune(7, 6) și Reuniune(2, 9).

B. În care caz nu se modifică vectorul T pe parcursul ciclului din procedură?

C. Cum se poate corecta procedura astfel încât, în cazul de la punctul *B*, să obținem o soluție mai rapidă?

P070117: Mulțime multiplă

(25 de puncte pentru categoria a II-a, 19 puncte pentru categoria a III-a)

Doi crescători de animale țin evidența animalelor în ordine alfabetică. Din fiecare animal pot fi mai multe exemplare (de exemplu: *cal*, *cal*, *cal*, *capră*, *iepure*, *iepure*). Animalele se găsesc în vectorul A care conține N elemente, respectiv în vectorul B care conține M elemente.

```
algoritm Primul(N, A, M, B, K, C)
  i ← 1
  j ← 1
  K ← 0
  cât timp i ≤ N și j ≤ M execută
    dacă A_i < B_j atunci
      i ← i + 1
    altfel
```

```
      dacă A_i > B_j atunci
        j ← j + 1
    altfel
        K ← K + 1
        C_K ← A_i
        i ← i + 1
        j ← j + 1
    sfârșit dacă
  sfârșit dacă
 sfârșit cât timp
sfârșit algoritm

algoritm AlDoilea(N, A, M, B, K, C)
  i ← 1
  j ← 1
  K ← 0
  A_N+1 ← "zzzzz"
  B_M+1 ← "zzzzz"
  cât timp i ≤ N + 1 și j ≤ M + 1 execută
    dacă A_i < B_j atunci
      K ← K + 1
      C_K ← A_i
      i ← i + 1
    altfel
      dacă A_i > B_j atunci
        K ← K + 1
        C_K ← B_j
        j ← j + 1
    altfel
        K ← K + 1
        C_K ← A_i
        i ← i + 1
        j ← j + 1
    sfârșit dacă
  sfârșit dacă
 sfârșit cât timp
sfârșit algoritm
```

A. Care va fi valoarea lui K și ce va conține vectorul C pentru fiecare algoritm, dacă avem N = 6, A = (cal, capră, capră, capră, iepure, iepure), M = 5, B = (cal, capră, capră, măgar, măgar)?

B. Care ar fi rezultatul celui de-al doilea algoritm pentru datele de mai sus, dacă condiția din ciclu ar fi (i ≤ N și j ≤ M)?

C. Explicați, în cazul celor doi algoritmi, care elemente vor fi inserate în vectorul C!

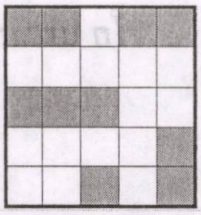
D. Pentru ce valori ale vectorilor A și B rezultatele generate de cei doi algoritmi vor fi aceleași?

P070118: Imagine

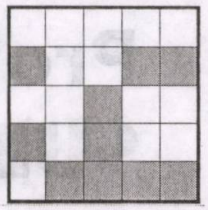
(21 de puncte pentru categoria a II-a, 15 puncte pentru categoria a III-a)

Definim următoarele operații de rotație în cazul imaginii de dimensiuni 5 × 5 (fiecare punct se deplasează într-o anumită direcție, iar cele care părăsesc imaginea, intră în imagine pe partea opusă):

- B: rotație în stânga;
- L: rotație în jos;
- J: rotație în dreapta;
- F: rotație în sus.

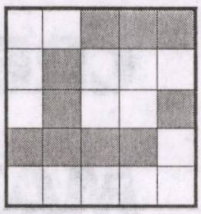


A. În minimum câte mutări putem obține din prima imagine cea de-a doua imagine, respectiv cea de-a treia imagine?



B. Dați un astfel de șir minim de mutări pentru cele două cazuri!

C. Dacă din prima imagine obținem câte o imagine prin operațiile C_1: JJJLLBJLBBF, C_2: JJJJLLFJJJF, respectiv C_3: LLJJLLJJ, care ar putea fi șirul minim, care dă același rezultat?



P070119: Optimizare

(25 de puncte, numai pentru categoria a III-a)

Fie următorul program, care descompune un număr dat, mai mic decât 10000 într-o sumă de pătrate de numere prime, în toate modalitățile posibile (de exemplu, $87 = 2^2 + 3^2 + 5^2 + 7^2$).

Modificați (optimizați) următorul program, oriunde credeți că este necesar, astfel încât să se execute mai rapid.

```
function prim(x: Longint): Boolean;
var q: Byte;
begin
  if x < 2 then
  begin
    prim := False;
    Exit
  end;
  for q := 2 to x - 1 do
    if x mod q = 0 then
    begin
      prim := False;
      Exit
    end;
  prim := True
end;

procedure back(sp: Byte; nr: Longint);
var q: Longint;
begin
  if (sp > 4) then
    if nr = 0 then
  begin
      for q := 1 to 4 do
        Write(ss[q],' ');
    WriteLn
  end else
  else
    for q := 1 to nr do
      if ((sp = 1) or (sp > 1) and
          (q >= ss[sp - 1])) and prim(q) then
      begin
        ss[sp] := q;
        back(sp + 1, nr - q * q)
      end
end;
```

Traducerea: Pătcaș Csaba

Problema SUBȘIRULUI CRESCĂTOR de LUNGIME MAXIMĂ și APLICAȚII ale acesteia

Dan Pracsiu

Problema determinării subșirului crescător de lungime maximă este una binecunoscută, printre primele care se studiază de obicei la programarea dinamică. Este prezentă în literatura de specialitate sub denumirea de Longest Increasing Substring (LIS). Ne propunem în acest articol să analizăm această problemă împreună cu alte câteva derivate din aceasta și care admit rezolvări strâns legate de rezolvarea problemei de bază.

În toate problemele care urmează se va face referire la un un șir $x = x_1, x_2, ..., x_n$, format din numere întregi, care este memorat într-un tablou unidimensional de lungime n. Nu vom face precizări suplimentare legate de valoarea lui n și nici despre intervalul de valori în care se încadrează fiecare element x_i al șirului. Motivul este dat de faptul că, în practică (viața reală), nu întotdeauna datele unei probleme sunt bine precizate de la început.

Problemele pe care le vom prezenta în cadrul acestui articol sunt:

- Să se determine lungimea maximă a unui subșir crescător din șirul x.
- Să se determine numărul subșirurilor crescătoare de lungime maximă.
- Să se determine subșirul crescător de lungime maximă, cel mai mic din punct de vedere lexicografic.
- Să se determine lungimea subșirului crescător de lungime minimă.
- Să se determine numărul de subșiruri crescătoare din șir.
- Se dau două șiruri A și B, ambele de lungime n, cu n par. Se știe că în ambele șiruri fiecare număr de la 1 la $n / 2$ apare de exact două ori. Să se determine lungimea maximă a unui subșir comun al celor două șiruri.

Vom analiza pe rând cele șase probleme, precizând mai întâi două noțiuni. Un subșir se obține din șirul inițial eliminând zero sau mai multe elemente, nu neapărat aflate pe poziții consecutive. Prin subsecvența de forma $x[i .. j]$ vom înțelege elementele șirului x aflate pe pozițiile de la i la j în șir, adică $x_i, x_{i+1}, ..., x_j$.

Lungimea maximă

Determinarea lungimii maxime se realizează cu ajutorul unui vector LIS de lungime n, în care $LIS[i]$ este lungimea maximă a unui subșir obținut din subsecvența $x[i .. n]$ și care conține elementul $x[i]$.

Obținem recurențele binecunoscute:

- $LIS[i] = 1$, dacă $x_i > x_j$ pentru $j \in i + 1 .. n$;

• $LIS[i] = 1 + \max(LIS[i] \mid j \in i+1 \,..\, n$ și $x_i \le x_j)$, în caz contrar.

Este evident că vom avea $LIS[n] = 1$. Soluția va fi dată de valoarea $\max\{LIS[i], i = 1 \,..\, n\}$

Algoritmul este următorul:

```
algoritm SubșirCrescătorDeLungimeMaximă
  LIS[n] ← 1
  pentru i ← n - 1, 1 pas -1 execută
    max ← 0
    pentru j ← i + 1, n execută
      dacă (max < LIS[i]) și (x[i] ≤ x[j])
                                    atunci
        max ← LIS[j]
      sfârșit dacă
    sfârșit pentru
    LIS[i] ← max + 1
  sfârșit pentru
  // determinarea lungimii maxime
  max ← LIS[1]
  for i ← 2, n execută
    dacă max < LIS[i] atunci
      max ← LIS[i]
    sfârșit dacă
  sfârșit pentru
  scrie max
sfârșit algoritm
```

Ordinul de complexitate este $O(n^2)$. Există, după cum se știe, algoritmul *Robinson-Schensted-Knuth*, al cărui ordin de complexitate este $O(n \cdot \log n)$, pe care nu îl vom analiza aici; un articol legat de acest algoritm a apărut în numărul din octombrie 2006 al *GInfo*.

Dacă se dorește nu doar determinarea lungimii maxime, ci și a unui subșir de lungime maximă, vom avea nevoie de un vector suplimentar *Next* de lungime n, care se construiește simultan cu *LIS* și în care, pentru fiecare i între 1 și n, în $Next[i]$ se va reține indicele j pentru care s-a determinat maximul.

Numărul șirurilor de lungime maximă

Vom construi, pe lângă vectorul *LIS*, un vector *NR* de lungime n, în care $NR[i]$ este numărul de subșiruri crescătoare de lungime maximă care se pot construi din secvența $x[i \,..\, n]$ și care încep cu $x[i]$.

Evident, avem $NR[n] = 1$, iar pentru fiecare i de la $n - 1$ la 1, valoarea $NR[i]$ va fi egală cu suma acelor valori $NR[j]$ pentru care $j > i$, $LIS[j] = LIS[i] - 1$ și $x[i] \le x[j]$.

Considerând că s-a obținut lungimea maximă *Lmax* a unui subșir crescător, soluția va fi dată de suma acelor valori $NR[i]$ pentru care $LIS[i] = Lmax$.

Evident, ordinul de complexitate al algoritmului este tot $O(n^2)$, cu mențiunea că este posibil ca soluția să fie un număr care nu se poate reprezenta pe 32 de biți, adică va fi nevoie de operații cu numere mari.

```
algoritm NumărSubșiruriCrescătoareDeLungime-
                                    Maximă
  // se determină vectorul LIS, la fel ca în cazul primei
  // probleme
  for i ← 1, n - 1 execută
    NR[i] ← 0
  sfârșit pentru
  NR[n] ← 1
  pentru i ← n - 1, 1 pas -1 execută
    L ← LIS[i]
    pentru j ← i + 1, n execută
      dacă x[i] ≤ x[j] și LIS[j] = L atunci
        NR[i] ← NR[i] + NR[j]
      sfârșit dacă
    sfârșit pentru
  sfârșit pentru

  // se determină în continuare
  //          Lmax = max{LIS[i], i = 1 .. n}
  S ← 0
  pentru i ← 1, n execută
    dacă LIS[i] = Lmax atunci
      S ← S + NR[i]
    sfârșit dacă
  sfârșit pentru
  scrie S
sfârșit algoritm
```

Cel mai mic șir maximal

Determinarea subșirului maximal, cel mai mic din punct de vedere lexicografic, poate fi realizată utilizând numai vectorul *LIS*, construit așa cum s-a arătat în cadrul prezentării primei probleme.

Ideea de rezolvare este următoarea: considerând *Lmax* ca fiind lungimea maximă a unui subșir, se caută poziția i_1 cu $x[i_1]$ minim și $LIS[i_1] = Lmax$; se caută apoi un indice $i_2 > i_1$ cu $x[i_2]$ minim și $LIS[i_2] = Lmax - 1$; procedeul continuă efectuându-se *Lmax* pași, la fiecare pas p ($p = 1 \,..\, Lmax$) căutându-se un indice $i_p > i_{p-1}$ cu $x[i_p]$ minim și $LIS[i_p] = Lmax - p + 1$; soluția este dată de $x[i_1], x[i_2], \ldots, x[i_{Lmax}]$; evident, aceste valori sunt ordonate crescător, lucru care se poate demonstra ușor.

```
algoritm SubșirMaximalLexicografic Minim
  // se consideră determinate vectorul LIS, precum și
  // lungimea Lmax
  poz ← 1
  pentru k ← Lmax, 1 pas -1 execută
    min ← ∞
    pentru i ← poz, n - k + 1 execută
      dacă min > x[i] și LIS[i] = k atunci
        p ← i
        min ← x[i]
      sfârșit dacă
    sfârșit pentru
    poz ← p
```

```
    scrie x[p]
  sfârșit pentru
sfârșit algoritm
```

Cel mai scurt subșir crescător

Problema determinării celui mai scurt subșir crescător (*SIS - Shortest Increasing Substring*) presupune un lucru evident: subșirul minimal nu mai poate fi extins.

Cu alte cuvinte, între oricare două elemente consecutive x_p și x_q (evident, $x_p \le x_q$) nu se poate adăuga un element x_k din șirul inițial cu proprietatea $p < k < q$ și $x_p \le x_k \le x_q$.

Spunem că subșirul de lungime minimă este maximal din punctul de vedere al proprietății de monotonie.

De exemplu, pentru șirul inițial 3, 1, 7, 2, 9, 5, subșirul crescător maximal de lungime minimă este 3, 5 și este unic. Subșirul 1, 9 nu este maximal de lungime minimă. El poate fi extins cu valoarea 7, obținându-se subșirul 1, 7, 9 și are lungimea 3. La fel, subșirul 2, 9 nu este maximal, el poate fi extins prin 1, obținându-se subșirul 1, 2, 9.

Pentru rezolvare, vom considera vectorul *SIS* de lungime n, unde *SIS*[i] va fi lungimea minimă a unui subșir maximal obținut din subsecvența $x[i \,..\, n]$ și care se începe cu $x[i]$. Recurența este una destul de asemănătoare cu cea de la determinarea celui mai lung subșir crescător:

- $SIS[i] = 1$, dacă $x_i > x_j$ pentru $j \in i + 1 \,..\, n$;
- $SIS[i] = 1 + \max(SIS[i] \mid j \in i + 1 \,..\, n, x_i \le x_j$ și nu există niciun k între $i + 1$ și $j - 1$ astfel încât $x_i \le x_k \le x_j)$, în caz contrar.

Este evident că vom avea $SIS[n] = 1$, iar soluția va fi dată de $\min\{SIS[i], i = 1 \,..\, n\}$. Condiția suplimentară care apare în determinarea minimului, anume că nu există k între $i + 1$ și $j - 1$ cu $x[i] \le x[k] \le x[j]$, este introdusă pentru a se evita obținerea unor subșiruri care nu sunt maximale.

Se poate determina ușor și subșirul minimal, nu doar lungimea sa, construind vectorul suplimentar *NEXT* de lungime n, în care *NEXT*[i] va memora valoarea j pentru care s-a determinat minimul, sau 0 (zero) dacă acest minim nu există, adică atunci când *SIS*[i] este 1.

Algoritmul, al cărui ordin de complexitate este pătratic, va fi prezentat în continuare.

```
algoritm CelMaiScurtSubșirMaximal
  SIS[n] ← 1
  pentru i ← n - 1, 1 pas -1 execută
    min ← ∞
    poz ← n + 1
    val ← ∞
    pentru j ← i + 1, n execută
      dacă min < SIS[j] și x[i] ≤ x[j] și
                              x[j] < val atunci
        min ← SIS[j]
        poz ← j
        val ← x[j]
      sfârșit dacă
    sfârșit pentru
    dacă poz ≤ n atunci
      SIS[i] ← 1 + min
    altfel
      SIS[i] ← 1
    sfârșit dacă
  sfârșit pentru
  min ← SIS[1]
  pentru i ← 2, n execută
    dacă min > SIS[i] atunci
      min ← SIS[i]
    sfârșit dacă
  sfârșit pentru
  scrie min
sfârșit algoritm
```

Pentru cunoscătorii teoriei grafurilor menționăm că șirului x i se poate asocia un digraf cu n vârfuri, corespunzătoare celor n componente ale șirului inițial x.

Vom construi arce de la un nod i la un nod j dacă $i < j$, $x[i] \le x[j]$ și nu există $k \in i + 1 \,..\, j - 1$ cu $x[i] \le x[k] \le x[j]$. Evident digraful obținut este aciclic.

Subșirul maximal de lungime minimă este, de fapt, un drum de lungime minimă de la un vârf de grad interior 0 la un vârf de grad exterior 0 (unde gradul interior al unui vârf i este dat de numărul de arce care "intră" în i, iar gradul exterior al unui vârf i este dat de numărul de arce care "ies" din i).

Numărul subșirurilor crescătoare

Dacă privim problema din punctul de vedere al teoriei grafurilor, atunci problema cere determinarea numărului de drumuri distincte de la vârfurile de grad interior 0 la nodurile de grad exterior 0 în digraful obținut asemănător ca la problema precedentă.

Vom prefera o rezolvare care să fie și pe înțelesul celor care nu au cunoștințe de grafuri. Construim tabloul unidimensional *NR* cu n componente, în care *NR*[i] este numărul de subșiruri crescătoare care există în subsecvența $x[i \,..\, n]$ și care încep cu $x[i]$.

Evident, $NR[n] = 1$, și pentru fiecare $i = n - 1 \,..\, 1$, *NR*[i] va fi egal cu suma acelor valori *NR*[j] cu proprietatea că $j \in i + 1 \,..\, n$, $x[i] \le x[j]$ și nu există $k \in i + 1 \,..\, j - 1$ astfel încât $x[i] \le x[k] \le x[j]$.

Observați condițiile care sunt preluate de la determinarea numărului de subșiruri maximale crescătoare și de la determinarea unui subșir crescător de lungime minimă.

Soluția se obține însumând acele valori *NR*[i] cu proprietatea că $x[i] < x[j]$, $\forall j \in 1 \,..\, i - 1$. Evident, *NR*[1] apare mereu în această sumă.

```
algoritm NumărSubșiruriCrescătoare
  pentru i ← 1, n - 1 execută
    NR[i] ← 0
  sfârșit pentru
  NR[n] ← 1
  max ← x[n]
```

```
pentru i ← n - 1, 1 pas -1 execută
    dacă x[i] > max atunci
      max ← x[i]
      NR[i] ← 1
    altfel
      val ← ∞
      pentru j ← i + 1, n execută
        dacă x[i] ≤ x[j] și x[j] ≤ val atunci
          val ← x[j]
          NR[i] ← NR[i] + NR[j]
        sfârșit dacă
      sfârșit pentru
    sfârșit dacă
  sfârșit pentru
  // determinăm soluția
  S ← NR[1]
  pentru i ← 2, n execută
    dacă min > x[i] atunci
      min ← x[i]
      S ← S + NR[i]
    sfârșit dacă
  sfârșit pentru
  scrie S
sfârșit algoritm
```

Ordinul de complexitate este $O(n^2)$, cu mențiunea că este posibil ca rezultatul să nu poată fi reprezentat pe 32 de biți, în acest caz fiind nevoie de operații cu numere mari.

Subșir comun

Cea de-a șasea problemă reprezintă o particularizare a problemei *Longest Common Subsequence* (*LCS*), care cere determinarea celui mai lung subșir comun a două șiruri *A* și *B*, de lungimi *n*, respectiv *m*. Ordinul de complexitate al algoritmului *LCS* este $O(n \cdot m)$.

Pentru cazul particular al acestei probleme, există algoritmul *Hunt-Szymanski* care extinde ideea de rezolvare în timp $O(n \cdot \log n)$ a primei probleme, obținându-se un ordin de complexitate $O(R \cdot \log n)$, unde *R* este numărul de potriviri, adică numărul de apariții ale elementelor șirului *A* în șirul *B*. De fapt, o potrivire este o pereche (p, q) pentru care $A_p = B_q$. Deoarece fiecare element din *A* apare de două ori în *B*, atunci numărul de potriviri este $R = 2 \cdot n$.

Să considerăm următorul exemplu: $n = 8$, $A = 4, 1, 3, 4, 2, 1, 2, 3$ și $B = 1, 3, 1, 2, 4, 3, 4, 2$. Potrivirile sunt următoarele: (1, 5), (1, 7), (2, 1), (2, 3), (3, 2), (3, 6), (4, 5), (4, 7), (5, 4), (5, 8), (6, 3), (6, 1), (7, 8), (7, 4), (8, 6), (8, 2). Vom ordona potrivirile crescător după pozițiile din șirul *A* și apoi descrescător după pozițiile din șirul *B*. Ordinea obținută va fi următoarea: (1, 7), (1, 5), (2, 3), (2, 1), (3, 6), (3, 2), (4, 7), (4, 5), (5, 8), (5, 4), (6, 3), (6, 1), (7, 8), (7, 4), (8, 6), (8, 2). Construim vectorul *LCS* de lungime *n*, în care *LCS*[*i*] va reține cel mai mic indice din vectorul *B* până la care se poate obține un subșir comun de lungime *i*.

Se consideră pe rând cele *R* potriviri, iar în vectorul *LCS* vor fi memorați, cum am spus, doar indici din *B*. La fel ca în cazul rezolvării liniarlogaritmice a problemei *LIS*, valorile din *LCS* vor apărea întotdeauna sortate crescător.

Pentru fiecare potrivire (p, q) se realizează o căutare binară pentru *q* în *LCS*[1 .. *max*], unde *max* este numărul de componente ocupate în vector, determinându-se o poziție *i* cu proprietatea $LCS[i - 1] < q < LCS[i]$. În acest caz, s-a găsit un indice *q* care este extremitatea dreaptă a unui subșir comun de lungime *i*, deci *LCS*[*i*] va reține valoarea *q*.

Dacă $q > LCS[max]$, atunci subșirul comun se extinde cu 1, iar în noua componentă va fi depusă valoarea *q*.

În tabelul de mai jos este descris modul de completare al vectorului *LCS* pentru exemplul dat anterior:

Pas	Potrivire	LCS
1.	(1, 7)	7
2.	(1, 5)	5
3.	(2, 3)	3
4.	(2, 1)	1
5.	(3, 6)	1, 6
6.	(3, 2)	1, 2
7.	(4, 7)	1, 2, 7
8.	(4, 5)	1, 2, 5
9.	(5, 8)	1, 2, 5, 8
10.	(5, 4)	1, 2, 4, 8
11.	(6, 3)	1, 2, 3, 8
12.	(6, 1)	1, 2, 3, 8 (*neschimbat*)
13.	(7, 8)	1, 2, 3, 8 (*neschimbat*)
14.	(7, 4)	1, 2, 3, 4
15.	(8, 6)	1, 2, 3, 4, 6
16.	(8, 2)	1, 2, 3, 4, 6 (*neschimbat*)

Subșirul comun maximal are lungimea 5. Ultima componentă ocupată din vectorul *LCS*, în exemplul nostru 6, ne indică și cel mai mare indice din *B* unde s-a găsit ultimul element comun celor două subșiruri.

După cum se observă din exemplu, algoritmul are exact $R = 2 \cdot n$ pași, pentru fiecare pas realizându-se o căutare binară. Vectorul *LCS* are, evident, lungimea maximă *n*. Așadar, numărul operațiilor este $2 \cdot n \cdot \log n$.

Să mai notăm faptul că sortarea potrivirilor nu adaugă timp suplimentar, deoarece se poate realiza în timp liniar. Lăsăm scrierea algoritmului ca exercițiu, considerând că exemplul dat este sugestiv.

Considerații finale

Apar destul de des în diverse concursuri probleme legate de determinarea subșirului crescător de lungime maximă sau variante; de exemplu, relația "≤" între elementele subșirului se poate schimba într-o relație oarecare. Este cazul problemei ***Lanț*** de la olimpiada județeană din 2005, clasele XI-XII. Cea de-a șasea problemă apare la barajul pentru stabilirea lotului *României* la *ONI* 2001; suplimentar, este cerută și determinarea efectivă a subșirului comun de lungime maximă.

Există de asemenea pe *site*-ul **http://infoarena.ro** și alte probleme legate de problema *LIS*. Una care tratează un alt caz particular al problemei *LCS* este problema ***Tabără*** de la etapa a treia a concursului *Campion*, ediția 2003.

Nu în ultimul rând, recomandăm cititorilor o altă problemă clasică, cea legată de determinarea celui mai lung subșir comun crescător (*Longest Common Increasing Subsequence* - *LCIS*), al cărui algoritm de rezolvare este cunoscut sub numele de *algoritmul lui Chao* și are o complexitate pătratică.

Cum împarți cinci pâini la trei călători?[1]

PARTIȚIILE de LUNGIME FIXĂ ale unui NUMĂR natural

Tiberiu Socaciu

Practic, pentru a rezolva o problemă cu ajutorul metodei Backtracking[2], este necesar ca în primul rând să punem în evidență elementele definitoare sau generatoare soluției sub forma unui vector (X), pentru fiecare element din vector trebuind pusă în corespondență o mulțime M_i[3], deci și o valoare în vectorul P. Uneori, aceasta este cea mai grea sau delicată problemă[4] cu care ne confruntăm. O alegere fericită a vectorului X permite o definire pentru ϕ[5], chiar dacă obținem un șir mare de condiții în conjuncție[6]. De regulă, prin serializare[7], ajungem la condițiile ω_k pe care le implementăm. Viteza de execuție a algoritmului obținut depinde de modul în care sunt implementate funcțiile Fi, precum și de eficiența detectării falselor soluții parțiale încă din fazele incipiente[8].

Ne propunem determinarea tuturor partițiilor de lungime *n* ale unui număr natural *m*.

Numim *partiție* de lungime *n* a unui număr natural *m* o descompunere[9] în *n* termeni pozitivi a numărului *m*.

De exemplu, toate partițiile distincte de lungime 3 ale lui 5 sunt:

```
5 = 1 + 1 + 3
5 = 1 + 2 + 2
5 = 1 + 3 + 1
5 = 2 + 1 + 2
5 = 2 + 2 + 1
5 = 3 + 1 + 1
```

Etapa #1: Modelarea problemei pentru a scrie soluția sub forma unui vector

Deoarece problema gravitează în jurul partiționării, soluția fiind chiar o partiție, vom identifica componentele vectorului soluție ca fiind chiar termenii care formează această partiție[10].

Etapa #2: Identificarea mulțimilor din care fac parte elementele vectorului soluție

Valoarea maximă pentru un termen se obține când toți ceilalți *n* - 1 termeni sunt minimi[11]. Din condiția ca suma lor să fie *m*, deducem că valoarea maximă este *m* - *n* + 1.

[1] *evident, aluzie la povestea lui Ion Creangă*
[2] *cu privire la metoda backtracking, vezi [1], [2], [3], [17]*
[3] *M_k este mulțimea suport pentru al k-lea element din vectorul soluție.*
[4] *de multe ori analiza unei probleme împreună cu modelarea matematică reprezintă etapele cele mai dificile în implementarea informatică a rezolvărilor*
[5] *funcția de continuare*
[6] *clauze legate prin operatorul "și"*
[7] *cu privire la serializare, vezi [1], [2], [3], [14]*
[8] *Cu privire la acest aspect, vezi "cea mai bună funcție de continuare" în [1], [2], [3], [13].*
[9] *descompunere privită ca sumă de termeni.*
[10] *$m = x_1 + x_2 + \ldots + x_n$*
[11] *au valoarea 1*

Așadar, toate mulțimile M_i sunt egale între ele și cuprind elemente mai mici decât $m - n + 2$, adică $p_1 = p_2 = ... = p_n = m - n + 1$.

Etapa #3: Identificarea functiei φ corespunzătoare

Din definiția partiției obținem condiția imediat: $\varphi(x) = "x_1 + x_2 + ... + x_n = m"$.

Etapa #4: Pregătirea funcției φ pentru serializare

Deoarece fiecare componentă a partiției este un număr natural nenul, rezultă că suma de forma $x_1 + x_2 + ... + x_k$ este mai mică decât m[12].

Dar ținem cont de recomandarea de la paragraful precedent și încercăm să fim mai exacți: $x_1 + x_2 + ... + x_k \le m - n + k$, pentru fiecare k cuprins între 1 și $n - 1$[13] și $x_1 + x_2 + ... + x_k = m$, pentru $k = n$[14].

Etapa #5: Scrierea și implementarea funcției de continuare

Prin serializare obținem funcțiile ω_k:

- $\omega_k(x_1, x_2, ..., x_k) = x_1 + x_2 + ... + x_k \le m - n + k$, pentru fiecare k cuprins între 1 și $n - 1$;
- $\omega_k(x_1, x_2, ..., x_k) = x_1 + x_2 + ... + x_k = m$, pentru $k = n$.

Prin urmare o posibilă implementare a funcției de continuare este:

```
function Fi(K: Integer; X: Sir): Boolean;
var I, S: Integer;
begin
  S := 0;
  for I := 1 to K do
  begin
    S := S + X[I]
  end;
  if K = N then
    begin
      Fi := S = M
    end
  else
    begin
      Fi := S <= M - N + K
    end
end;
```

Etapa #6: Scrierea programului.

Programul se obține prin legarea funcției implementate de restul componentelor.

Implementarea este prezentată în casete.

[12] *deoarece* $x_{k+1} + x_{k+2} + ... + x_m \ge 0$

[13] *deoarece suma următorilor n - k termeni este minimă când fiecare termen ia valoarea 1*

[14] *pentru îndeplinirea condiției* $\varphi_n = \varphi$

Listing PARTITII.PAS (I)

```
Program GenPartitii;
const MaxN = 100;
          { dimensiunea maxima a partitiei }
type Sir = array[1..MaxN] of Integer;
                    { sir de numere intregi }
var M, N: Integer;

{ afisarea unei solutii determinate }
procedure PrelucreazaSolutie (N: Integer;
                              X: Sir);
var I: Integer;
begin
  Write (M, ' = ', X[1]);
  for I := 2 to N do
 begin
    Write(' + ', X [I])
 end;
  WriteLn
end;

{ functia de continuare pe baza serializarii }

function Fi(K: Integer; X: Sir): Boolean;
var I, S: Integer;
begin
  S := 0;
  for I := 1 to K do
  begin
    S := S + X[I]
  end;
  if K = N then
    begin
      Fi := S = M
    end
  else
    begin
      Fi := S <= M - N + K
    end
end;

{ determinarea partitiilor de lungime N    }
{ ale numarului M, folosind metoda         }
{ backtracking                             }
procedure Partitii(N, M: Integer);
var B: Boolean;
    K: Integer;
    X: Sir;
```

Listing PARTITII.PAS (I)

```
begin
  K := 1;
  X[1] := 0;

  while K > 0 do
  begin
    repeat
      X[K] := X[K] + 1;
      B := Fi(K, X) or
           (X[K] > M - N + 1)
    until B;

  if X[K] > M - N + 1 then
   begin
        K := K - 1
     end
    else
      begin
        if K = N then
          begin
            PrelucreazaSolutie(N, X)
          end
        else
          begin
            K := K + 1;
            X[K] := 0
          end
      end
  end
end;

{ programul principal }
Begin
  Write('Numarul de partitionat: ');
  ReadLn(M);
  Write('Dimensiunea partitiei: ');
  ReadLn(N);
  Partitii(N, M)
End.
```

Etapa #7: Verificarea programului

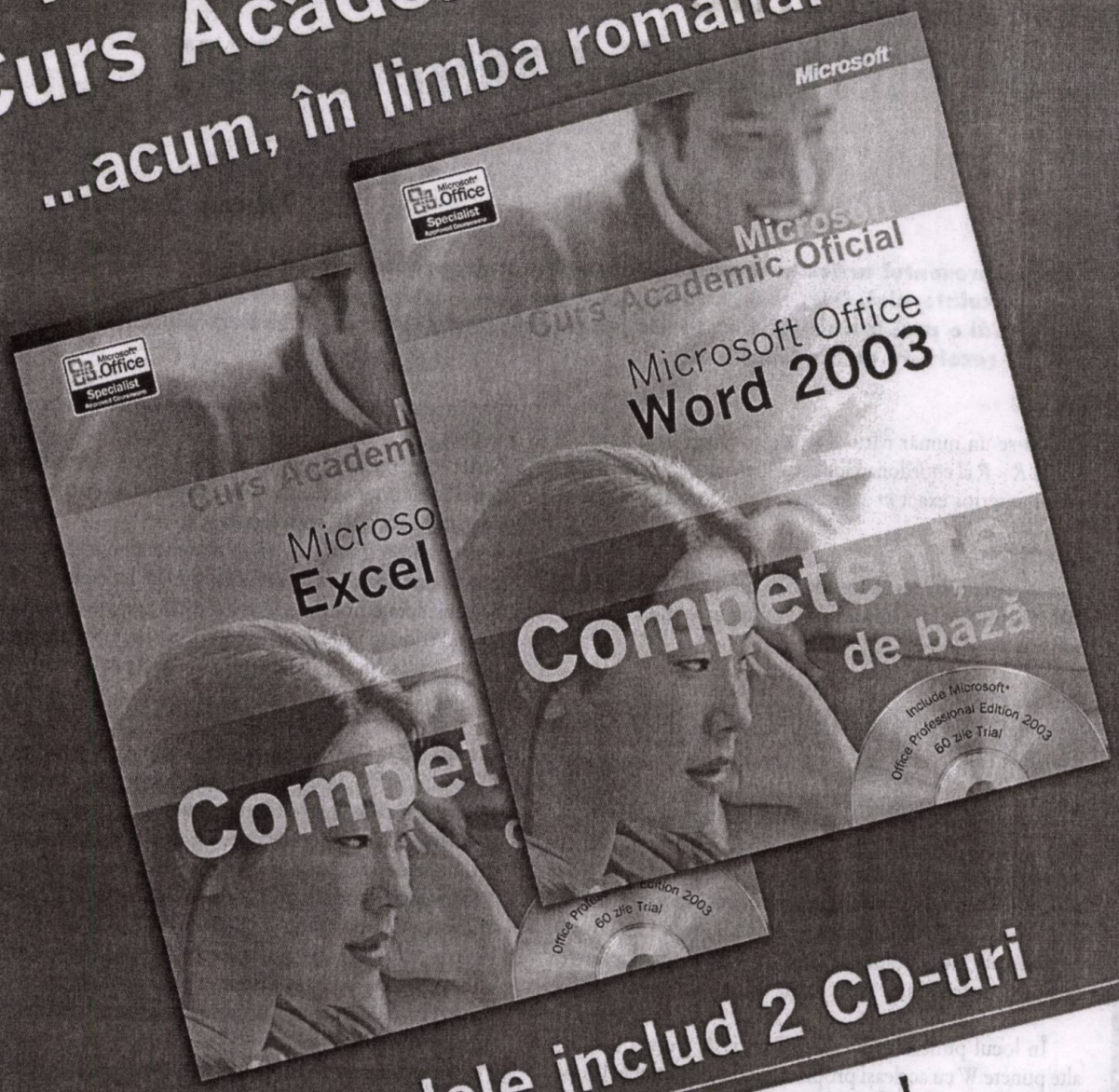
- cele mai frecvente erori sunt cele de compilare (cine nu tastează uneori în grabă **begn** în loc de **begin**?);
- urmează apoi greșeli de tastare care nu se văd la compilare (un `m` în loc de `n` este foarte greu de detectat);
- urmează apoi greșeli de introducere a datelor: lipsa unor mesaje pe ecran ne face uneori să greșim (cine își mai amintește că întâi îl citesc pe `m` și apoi pe `n` sau invers?);
- am lăsat la final greșelile de algoritm pentru că uneori sunt cel mai greu de reparat; dar, s-a demonstrat ca algoritmul implementat este corect .

Bibliografie

1. Tiberiu Socaciu, *Învățați Backtracking. Exemple în Pascal și C*, Editura InfoData Cluj 2006.
2. Tiberiu Socaciu, *Metoda Backtracking cu exemple în Pascal*, Editura Edusoft Bacău 2005.
3. Tiberiu Socaciu, *Metoda Backtracking cu exemple în C*, Editura Edusoft Bacău 2005.
4. Tiberiu Socaciu, *Informatica privită... altfel sau 16 chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft Bacău 2005.
5. Bogdan Pătruț, Tiberiu Socaciu, *Backtracking și Recursivitate. Culegere de Exercitii și Probleme*, în curs de apariție.
6. Tiberiu Socaciu, *Culegere de Exerciții și Probleme de Informatică*, în curs de apariție.
7. Tiberiu Socaciu, *Informatica privită... altfel (vol. 2)*, în curs de apariție.
8. Bogdan Pătruț, Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, comunicare la "Linux și mediile virtuale de instruire", Arad, 11 - 15 septembrie 2006, eveniment organizat de Universitatea de Vest "Vasile Goldiș" din Arad.
9. Darius Bufnea, Tiberiu Socaciu, *De la backtracking secvențial la cel concurent și paralel*, comunicare susținută la Zilele Unirii, Focșani, 22 - 24 ianuarie 2005.
10. Tiberiu Socaciu, *Generalizarea problemei turnurilor din Hanoi*, comunicare la Tabăra "Așii Informaticii", Piatra Neamț, iunie 1995.
11. Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, GInfo nr. 16/8 (decembrie 2006), Editura Agora Media, Târgu Mureș.
12. Bogdan Pătruț, Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, Studia Universitatis "Vasile Goldiș" Arad, Seria Științe Exacte, nr. 16 (2006), în curs de aparitie.
13. Tiberiu Socaciu, *Cea mai bună funcție de continuare*, GInfo nr. 16/7 (noiembrie 2006), Editura Agora Media, Târgu Mureș.
14. Tiberiu Socaciu, *Serializarea ca metodă de determinare a functiei de continuare*, GInfo nr. 16/6 (octombrie 2006), Editura Agora Media, Târgu Mureș.
15. Tiberiu Socaciu, *O problemă de combinatorică*, GInfo nr. 11/4 (aprilie 2001), Editura Agora Media, Târgu Mureș.
16. Tiberiu Socaciu, *O extindere a problemei Turnurilor din Hanoi*, Gazeta de Informatică, nr. 11-12/1994, Editura Libris, Cluj-Napoca.
17. Tiberiu Socaciu, *Metoda Backtracking*, în *if* nr. 5/1992 (16), Editura Computer Press Agora, Târgu Mureș.

NOU!

Microsoft®
Curs Academic Oficial
...acum, în limba română!



Manualele includ 2 CD-uri

Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Problema STEINHAUS-SIERPINSKI

Tiberiu Socaciu

În prezentul articol încercăm să rezolvăm câteva probleme cu ajutorul calculatorului. Dar, rezolvarea acestor probleme este simplă dacă studiem întâi o problemă clasică în teoria numerelor, problemă enunțată de Steinhaus și rezolvată de Sierpinski.

Enunț

Dându-se un număr natural m, determinați dacă există în planul $R \times R$ al coordonatelor carteziene un cerc care să conțină în interior exact m puncte care au coordonate întregi.

Soluție

Rezolvarea problemei a fost dată de celebrul *W. Sierpinski*, care a formulat următoarea propoziție care stă la baza rezolvării.

Propoziție

Distanțele de la $W(2^{1/2}, 1/3)$ la oricare punct din $Z \times Z$ sunt diferite între ele.

Demonstrație

Presupunem, prin absurd, că există în plan două puncte distincte $A(a_1, b_1)$ și $B(a_2, b_2)$ pentru care avem $d(A, W) = d(B, W)$. Pornind de la definiția euclidiană a distantei ajungem la o ecuație diofantică ce devine $a_2 + b_2 - 2 / 3 = 0$, ecuație care nu are soluții întregi. Așadar, presupunerea este falsă, prin urmare propoziția inițială este adevărată.

În locul punctului W al lui *Sierpinski* putem defini și alte puncte W cu aceleași proprietăti. De exemplu, în [Pope] se folosește punctul $W(2^{1/2}, 3^{1/2})$ în plan sau punctul $W(2^{1/2}, 3^{1/2}, 5^{1/2})$ în spațiul tridimensional, iar în [Scho] este studiată problema determinării mulțimii acestor puncte W oferindu-se și alte variante de alegere a acestora.

Probleme propuse spre rezolvare:

1. Scrieți un program care să determine un cerc care conține în interiorul său exact 2007 puncte laticiale.
2. Se consideră o mulțime formată din n puncte laticiale în plan. Scrieți un program care determină un cerc care conține în interiorul său exact m ($m < n$) dintre cele n puncte date.
3. Scrieți un program care determină două cercuri secante, astfel încât intersecția interioarelor celor două cercuri să conțină exact 2007 puncte laticiale.
4. Scrieți un program care să determine o sferă care conține în interiorul său exact 2007 puncte laticiale.
5. Se consideră o mulțime formată din n puncte laticiale în spațiul tridimensional. Scrieți un program care determină o sferă care conține în interiorul său exact m ($m < n$) dintre cele n puncte date.

Indicații

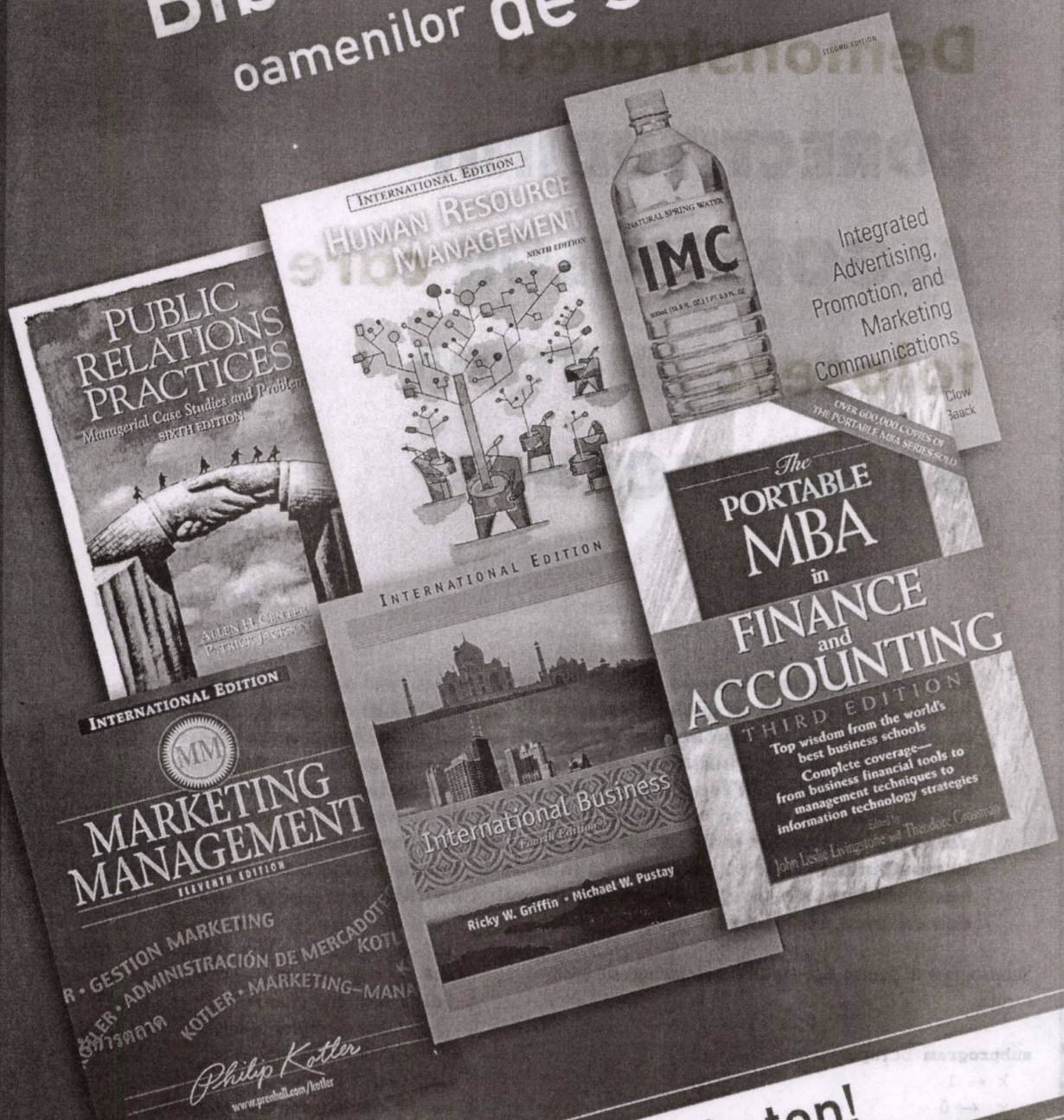
1. Deoarece pe post de puncte de tip *Sierpinski* sunt folosite puncte care au coordonate numere irationale, pentru a crește viteza de execuție, încercați să implementați o funcție de comparare a distanței de la acest punct la două puncte laticiale, în așa fel încât să nu fie necesară utilizarea de operații în virgulă mobilă.
2. Pentru problema 2, pentru a evita o prealabilă sortare a celor n puncte pe baza distanței până la W, încercați să construiți mulțimea finală a celor m puncte "din mers", printr-un algoritm liniar.
3. Pentru problema 3, încercați să folosiți cercul găsit la problema 1, iar cel de-al doilea cerc fixați-l cu centrul pe primul cerc, raza lui fiind diametrul primului cerc din care se scade un mic ecart.

Bibliografie

[Pope] **Popescu M., Bucur Ion,** *Puncte laticiale*, referat pentru Cercul Pedagogic de Matematică, 17 noiembrie 2005, **www.didactic.ro/files/3/t.e.s.t.4.doc.**

[Scho] **Isaac J. Schoenberg,** *Priveliști matematice*, Editura Tehnică București, 1989.

Biblioteca oamenilor de succes



Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Demonstrarea CORECTITUDINII ALGORITMILOR care folosesc METODA BACKTRACKING

Tiberiu Socaciu

Una dintre cele mai comode metode de rezolvare a unei probleme pentru care habar nu avem cum să găsim o soluție eficientă este backtracking-ul. Se știe că este un algoritm exponențial și este descurajant, dar argumentul pentru folosirea lui ar fi posibilitatea rulării acestuia pe un cluster de calculatoare, posibilitatea rafinării funcțiilor de continuare prin serializare spre a ne duce cât mai aproape de cea mai bună funcție de continuare (vezi bibliografia, o parte a materialelor publicate și in GInfo). Singura problemă care se mai pune în momentul în care ne lăsăm viața în mâinile acestui algoritm este: algortmul backtracking este corect? Ne rezolvă el problema? Aici vom incerca să demonstrăm acest lucru.

Subprogramul[1] pentru algoritmul *backtracking* este următorul[2]:

```
subprogram bt(np, p, x)
  k ← 1
  x_k ← 0
  cât timp k > 0 execută
    repetă
      incrementează(x_k)
    până când fi(k, x) sau x_k > p_k
    dacă x_k ≤ p_k atunci
      dacă k = np atunci
        // am identificat o soluție
        prelucrează_soluție(np, x)
      altfel
        // continuăm căutarea soluției
        incrementează(k)
        x_k ← 0
      sfârșit dacă
    altfel
      decrementează(k)
    sfârșit dacă
  sfârșit cât timp
sfârșit subprogram
```

[1] *subprogramul este descris într-un limbaj pseudocod, parametrii se transmit prin valoare*

[2] *o variantă, de reprezentare a acestuia*

Apelul subprogramului se realizează prin:

```
bt(np, p, x)
```

În cadrul secvenței anterioare (prezentarea generică în pseudocod a algoritmilor care folosesc metoda *backtracking*), `fi` este un subprogram de evaluare a funcției de continuare.

Vom demonstra mai întâi că orice rezultat furnizat de `bt` este corect, adică φ este adevărat.

Trebuie remarcat faptul că unicul loc în care are loc o ieșire a datelor este apelul `prelucrează_soluție`; dar, acest lucru nu poate să se întâmple decât dacă avem `k = np`, adică avem deja completate `np` valori din vectorul cu soluții parțiale `x`, valori care au trecut toate de testul de acceptanță prin apelul funcției de continuare `fi`, deci datele "scoase" reprezintă soluții.

Vom demonstra acum că nu există soluții pe care `bt` să nu le determine.

Presupunem, prin absurd, că vectorul $y_1, y_2, \ldots, y_{np}$ reprezintă o soluție[3] nedeterminată de `bt`.

Fie `k` maximul numărului de elemente consecutive, începând cu prima poziție, identice din vectorul `y` și vectorul soluțiilor parțiale `x` de-a lungul perioadei de viață a acestuia.

Evident, $fi(k, y_1, y_2, \ldots, y_k)$ este adevarat, deci după incrementarea valorii `k` se intră din nou în căutarea unui nou candidat, în urma incrementărilor succesive ale elementului x_k se va ajunge, la un moment dat, la soluția parțială pe `k + 1` elemente pentru care $x_{k+1} = y_{k+1}$.

De asemenea, este evident faptul că această soluție verifică și $fi(k + 1, y_1, y_2, \ldots, y_{k+1})$, adică avem `k + 1` valori identice, deci valoarea `k` aleasă nu este maximă cu proprietatea cerută, ceea ce reprezintă o contradicție.

Am demonstrat că algoritmul este parțial corect[4]; urmează să demonstrăm ca este și finit[5].

Fie `pmax` maximul elementelor vectorului `p`. Definim funcția:

$$W: \{1, 2, \ldots, np\} \times M \rightarrow N,$$

unde `W(K, X)` este numărul format din `np` cifre în baza `pmax`, cu primele `k` cifre din vectorul `x`, iar ultimele `np - k` cifre `0` pentru o solutie parțială la care nu se revine prin decrementare.

`W(K, X)` este definit la fel ca mai sus adăugându-se $0.5 \cdot pmax^k$. În mod evident `W` este o funcție limitată superior de valoarea $pmax^{NP+1}$.[6]

Rămâne de arătat faptul că funcția `W` este monoton crescătoare.

Acest lucru, însă, poate fi demonstrat imediat pe baza observației faptului că, în cadrul subprogramului `bt`, la fiecare iterație are loc o creștere a valorii x_k, implicit și a valorii funcției `W`, sau o creștere a valorii `W` cu $0.5 \cdot pmax^k$ dacă are loc decrementarea valorii `k`.

Prin urmare, am demonstrat că algoritmul este total corect[7].

Bibliografie

1. **Tiberiu Socaciu,** *Învățați Backtracking. Exemple în Pascal și C*, Editura InfoData Cluj-Napoca, 2006, ISBN 973-87774-1-0, 160 de pagini;
2. **Tiberiu Socaciu,** *Metoda Backtracking cu exemple în Pascal*, Editura Edusoft Bacău, 2005, ISBN 973-87496-5-4, 100 de pagini;
3. **Tiberiu Socaciu,** *Metoda Backtracking cu exemple în C*, Editura Edusoft Bacău, 2005, ISBN 973-87496-6-2, 100 de pagini;
4. **Bogdan Pătruț, Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, comunicare la Linux și mediile virtuale de instruire, Arad, 11-15 septembrie 2006, eveniment organizat de Universitatea de Vest "Vasile Goldiș" Arad;
5. **Darius Bufnea, Tiberiu Socaciu,** *De la backtracking secvențial la cel concurent și paralel*, comunicare susținută la Zilele Unirii, Focșani, 22-24 ianuarie 2005;
6. **Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, GInfo nr. 16/8 (decembrie 2006), Editura Agora Media Târgu-Mureș;
7. **Tiberiu Socaciu,** *Cea mai bună funcție de continuare*, GInfo nr. 16/7 (noiembrie 2006), Editura Agora Media Târgu-Mureș;
8. **Tiberiu Socaciu,** *Serializarea ca metoda de determinare a funcției de continuare*, GInfo nr. 16/6 (octombrie 2006), Editura Agora Media Târgu-Mureș;
9. **Tiberiu Socaciu,** *O problemă de combinatorică*, GInfo nr. 11/4 (aprilie 2001), Editura Agora Media Târgu-Mureș;
10. **Tiberiu Socaciu,** *Metoda Backtracking*, if nr. 5/1992 (16), Editura Computer Press Agora Târgu-Mureș;
11. **Bogdan Pătruț, Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, Studia Universitatis "Vasile Goldiș" Arad, seria Științe Exacte, nr. 16 (2006), în curs de apariție.
12. **Tiberiu Socaciu,** *Principiul contorului*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
13. **Tiberiu Socaciu,** *Rezolvarea unei probleme de backtracking*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
14. **Tiberiu Socaciu,** *Corectitudinea programelor*, teză de dizertație, Universitatea Babeș-Bolyai Cluj, Facultatea de Matematică și Informatică, 1996 (conducător științific: prof. univ. dr. Militon Frențiu).

[3] *se verifică φ(Y)*

[4] *produce toate rezultatele în ipoteza că se termină*

[5] *se termină într-un număr finit de pași*

[6] *explicația se bazează pe transcrierea valorii numărului dat de W*

[7] *parțial corect și finit*

COMBINATORICĂ în EXCEL

László Zsakó

La concursul de informatică al Revistei de Matematică, organizat pentru elevii din Ungaria, se propun spre rezolvare (și) probleme de combinatorică care pot fi rezolvate cu un tabelator. Vom prezenta câteva dintre aceste probleme, precum și rezolvările date de concurenți. Vom arăta că probleme tradiționale de matematică pot fi rezolvate cu ajutorul unor produse software care necesită, un mod de gândire diferit de cel necesar atunci când rezolvăm problema pe hârtie sau scriind un cod într-un limbaj de programare.

Se știe că un tabelator poate să conțină în celulele sale diferite date sau formule ale căror valoare se evaluează folosind valori aflate în alte celule. Astfel, cu ajutorul unui tabelator se pot completa tabele cu valori, cum ar fi *triunghiul lui Pascal*.

Este important să reținem că formulele respective nu pot conține referințe recursive, adică ele trebuie să definească o secvență de calcule.

Într-un tabelator orice formulă poate fi extinsă asupra unui domeniu rectangular, dar trebuie să le concepem astfel încât celulele corespunzătoare pozițiilor nedefinite să rămână vide.

Ca prim exemplu, am ales problema cea mai simplă și totodată cea mai cunoscută din tipul de probleme discutat: *triunghiul lui Pascal*. Să reținem că - în mod diferit de problemele tradiționale de programare (de algoritmică) unde, de regulă, se lucrează cu tablouri care se declară și sunt umplute cu valori determinate cu ajutorul formulelor date în specificații sau sunt deduse cu ajutorul unor structuri de control iterative - aici nu trebuie să ne preocupe ordinea în care se efectuează calculele.

Triunghiul lui Pascal

Aranjarea uzuală a coeficienților binomiali (în *triunghiul lui Pascal*) se poate vedea în imagine.

Se știe că acest tabel se completează calculând valoarea unui element (cu excepția celor de pe laturi) ca fiind suma dintre elementul situat deasupra lui, respectiv cel din stânga acestuia.

	A	B	C	D	E	F	G	H
1	1							6
2	1	1						
3	1	2	1					
4	1	3	3	1				
5	1	4	6	4	1			
6	1	5	10	10	5	1		
7	1	6	15	20	15	6	1	

Vom crea un tabel *Excel* cu primele $n + 1$ linii ale triunghiului lui *Pascal*. Valoarea lui n va fi scrisă în celula `H1` ($1 \le n \le 20$). Tabelul va avea exact $n + 1$ linii.

Știm că pentru orice n, avem:

- $B(n, 0) = 1$;
- $B(n, n) = 1$.

Ca urmare, fiecare celulă din coloana `A` și celulele `A1`, `B2`, `C3`, ... trebuie să conțină valoarea `1`.

Avem acum următoarea sarcină: să lăsăm vide celulele corespunzătoarea unor valori mai mari decât n.

Deoarece nu avem o variabilă de control, în formulă avem nevoie de indicele liniei în care se află celula curentă, dată de funcția `ROW()` a tabelatorului.

Liniile, de regulă, sunt numerotate de la 1, dar în triunghiul lui *Pascal* ele trebuie numerotate de la 0, deci vom completa liniile 1, 2, ..., $n + 1$, aplicând formula:

```
=IF(ROW()<=$H$1+1,1,"").
```

Pentru restul celulelor avem una dintre următoarele posibilități:

- $B(n, k) = B(n - 1, k) + B(n - 1, k - 1)$;
- $B(n, k) = n / k \cdot B(n - 1, k - 1)$;
- $B(n, k) = n / (n - k) \cdot B(n - 1, k)$.

Folosind prima formulă, aceasta poate fi scrisă în celula `B3` astfel:

```
=IF(ROW()<=$H$1+1,A2+B2,"").
```

Ar urma extinderea acestei formule în interiorul triunghiului, ceea ce nu este chiar simplu. De asemenea, este dificil să producem valori 1 pe diagonală, cu toate că există modalitatea de a modifica formulele extinse, astfel încât să avem "rezultate" corecte și pe diagonală și deasupra diagonalei.

Prezentăm rezolvarea corectă, care folosește o formulă care conține o alternativă triplă (în interiorul triunghiului, pe diagonală, deasupra diagonalei):

```
=IF(ROW()<=$H$1+1,
  IF(ROW()>COLUMN(),A2+B2,
    IF(ROW()=COLUMN(),1," "))).
```

Aplicație cu coeficienții binomiali

La un meci, atunci când echipa lor câștigă, n suporteri își aruncă în sus șepcile.

Apoi, când acestea cad înapoi, fiecare persoană prinde exact o șapcă.

Vom afișa numărul cazurilor în care dintre cei n ($0 \le n \le 12$) suporteri exact k ($0 \le k \le n$) persoane își vor prinde propriile șepci.

În exemplu avem n și $k \le 4$.

	A	B	C	D
1	1			
2	0	1		
3	1	0	1	
4	2	3	0	1

Pentru a prezenta rezolvarea, avem nevoie de explicații din combinatorică.

Soluția problemei se calculează ca fiind produsul a două numere.

Primul este numărul modalităților în care se pot selecta k elemente dintre n elemente - acestea sunt coeficienții binomiali notați prin $B(n, k)$.

Celălalt este numărul modalităților în care $n - k$ elemente se pot permuta astfel încât niciunul dintre ele să nu rămână în poziția sa originală - acesta este $S(n - k)$, numărul permutărilor dispersate a $n - k$ elemente.

Astfel se conturează rezolvarea: $H(n, k) = B(n, k) \cdot S(n - k)$, expresie în care primul factor se calculează exact ca în prima problemă.

Calculul celui de-al doilea factor este puțin mai complicat.

Avem:

$$H(n, 0) + H(n, 1) + ... + H(n, n) = n!$$

De aici deducem:

$$n! = B(n, 0) \cdot S(n - 0) + B(n, 1) \cdot S(n - 1) + ... + B(n, n) \cdot S(n - n).$$

Știm că:

$$B(n, k) = B(n, n - k).$$

Obținem astfel:

$$n! = B(n, n - 0) \cdot S(n - 0) + B(n, n - 1) \cdot S(n - 1) + ... + B(n, n - n) \cdot S(n - n).$$

Această formulă poate fi rescrisă sub forma:

$$n! = B(n, 0) \cdot S(0) + B(n, 1) \cdot S(1) + ... + B(n, n) \cdot S(n).$$

De aici obținem:

$$S(n) = (-1)^n \cdot (B(n, 0) \cdot (-1)^0 \cdot 0! + B(n, 1) \cdot (-1)^1 \cdot 1! + ... + B(n, n) \cdot (-1)^n \cdot n!) \quad [1].$$

Să vedem cum lucrăm cu tabelatorul: vom permite valori $n \le 12$, chiar dacă astfel vom întâlni și numere mari.

Fie `Bin` denumirea foii de lucru care calculează valorile $B(n, k)$ conform rezolvării primei probleme.

Deschidem o altă foaie de lucru, denumită `Work`, unde vom calcula factorialii (în coloana `A`), și puterile lui `-1` (în coloana `B`).

În celulele `A1` și `B1` scriem 1, corespunzător valorilor `0!` și -1^0.

În celula `A2` scriem formula:

```
=A1*(ROW()-1).
```

În celula `B2` scriem formula:

```
=-B1.
```

Extindem aceste formule în jos la dimensiunea cerută (13 linii).

În foaia `Disp` vom calcula valorile de care avem nevoie în suma numerelor permutărilor dispersate, unde valoarea celulei `A1` este:

```
=Bin!A1*
  INDEX(Work!$A$1:Work!$A$13,COLUMN(),1)*
  INDEX(Work!$B$1:Work!$B$13,COLUMN(),1).
```

Extindem această formulă de-a lungul a 12 linii (mai întâi extindem și triunghiul lui *Pascal* de pe foaia `Bin` pe 13 linii).

Acum calculăm valoarea lui $S(n)$ în coloana `C` a foii de calcul `Work`.

Fie valoarea lui C1:

```
=SUM(OFFSET(Disp!$A$1:Disp!$M$13,
                    ROW()-1,0,1,ROW()))*B1.
```

Această formulă o trecem peste coloana C.

Valorile $H(n, k)$ le calculăm prin înmulțirea a două valori din foaia principală.

Conținutul celulei A1 în această foaie este dat de formula:

```
=Bin!A1*OFFSET(Work!$C$1:Work!$C$13,
                    ROW()-COLUMN(),0,1,1).
```

Extindem această formulă pe domeniul A1..Z13 al foii de calcul.

Mai avem un singur lucru de făcut, și anume (la fel ca pentru prima problemă) să creăm celule vide în pozițiile care corespund unui indice mai mare decât n.

Triunghiul numerelor lui Euler

Triunghiul numerelor lui *Euler* este similar cu triunghiul lui *Pascal*, dar în locul coeficienților binomiali, acesta conține numerele lui *Euler* $E(n, k)$.

Pentru $0 \le n \le 15$ și $0 \le k \le n$, $E(n, k)$ este definit ca fiind numărul permutărilor mulțimii $\{1, 2, ..., n\}$, având exact k secvențe crescătoare (adică k perechi de poziții adiacente în permutare care au valori crescătoare).

Vom pregăti o foaie de calcul care - având ca intrare valoarea lui n în celula A1 - afișează numerele lui *Euler* $E(n, k)$ în primele $n + 1$ linii ale foii. Vom afișa numere doar în celule valide.

	A	B	C	D	E	F	G	H	I	J	K
1	1										
2	1	0									
3	1	1	0								
4	1	4	1	0							
5	1	11	11	1	0						
6	1	26	66	26	1	0					
7	1	57	302	302	57	1	0				
8	1	120	1191	2416	1191	120	1	0			
9	1	247	4293	15619	15619	4293	247	1	0		
10	1	502	14608	88234	156190	88234	14608	502	1	0	
11	1	1013	47840	455192	1310354	1310354	455192	47840	1013	1	0

Sarcina noastră este de a descoperi similitudini cu triunghiul lui *Pascal*, și bazându-ne pe acestea, de a scrie o rezolvare similară cu cea dată pentru prima problemă.

Numărul total de permutări a n elemente, având exact k perechi crescătoare poate fi determinat în două moduri pe baza numărului total de permutări a $n - 1$ elemente.

Una dintre modalități calculează numărul permutărilor de $n - 1$ elemente, unde există exact k perechi crescătoare. Dacă așezăm cel de-al n-lea număr într-o pereche crescătoare, numărul perechilor crescătoare rămâne k. Acest număr rămâne neschimbat, chiar dacă așezăm cel de al n-lea număr la începutul secvenței.

Cealaltă modalitate este de a porni de la numărul permutărilor a $n - 1$ elemente, unde există exact $k - 1$ permutări crescătoare.

Aici numărul permutărilor crescătoare crește cu 1, dacă al n-lea element este așezat la sfârșitul secvenței.

De asemenea, numărul permutărilor crescătoare crește, dacă al n-lea număr este așezat într-o poziție unde nu era asigurată creșterea.

Așadar, soluția este:

$$E(n, k) = (k + 1) \cdot E(n - 1, k) + (n - k) \cdot E(n - 1, k - 1).$$

Valoarea lui n va fi scrisă în celula A1, iar în celula A2 va fi scrisă formula:

```
=IF(ROW()<=$A$1+2,ROW()-2,"").
```

Această va fi extinsă în coloana A pe maximum 15 linii conform cerinței problemei. (Se poate scrie o formulă similară și în linia 1.)

Astfel, pentru $k = 0$, avem $E(n, k) = 1$ (există o singură secvență monoton descrescătoare), deci în celula B2 scriem formula:

```
=IF(A2="","",1).
```

Vom umple coloana B cu această formulă.

Pentru celelalte coloane lăsăm zona de deasupra triunghiului vidă; bineînțeles, pe diagonală ($E(n, n)$) trebuie să avem 0, iar celelalte elemente se pot calcula din linia precedentă cu ajutorul formulei de mai sus.

În celula C3 scriem formula de mai jos, pe care apoi o extindem:

```
=IF(OR(COLUMN()>ROW(),B3=""),"",
  IF(COLUMN()=ROW(),0,(COLUMN()-1)*C2+
                    (ROW()-COLUMN())*B2)).
```

Extinderea triunghiului lui Pascal

Coeficienții binomiali din triunghiul lui *Pascal* ($B(n, k)$) pot fi determinați și pentru valori negative ale lui n. Vom crea un tabel cu coeficienții binomiali calculați pentru un n negativ dat, de la $B(0, 0)$ la $B(-n, n)$.

	A	B	C	D	E	F
1	1	-6	21	-56	126	-252
2	1	-5	15	-35	70	-126
3	1	-4	10	-20	35	-56
4	1	-3	6	-10	15	-21
5	1	-2	3	-4	5	-6
6	1	-1	1	-1	1	-1
7	1	0	0	0	0	0

Avem relația cunoscută:

$$B(i + 1, k) = B(i, k) + B(i, k - 1).$$

Vom rearanja această formulă înlocuind valoarea lui i cu $-n$:

$$B(-n + 1, k) = B(-n, k) + B(-n, k - 1).$$

În coloana B scriem 1, în linia 13 scriem 0, iar în celula C12 scriem formula =C13-B12 pe care o extindem.

Triunghiul de speța a II-a al lui Euler

Vom crea o foaie de calcul care afișează triunghiul numerelor de speța a II-a ale lui *Euler* (numerele ($E(n, k)$) în $n + 1$ linii. Valoarea lui n va fi scrisă în celula A1.

	A	B	C	D	E	F	G	H
1	1							
2	1	0						
3	1	2	0					
4	1	8	6	0				
5	1	22	58	24	0			
6	1	52	328	444	120	0		
7	1	114	1452	4400	3708	720	0	
8	1	240	5610	32120	58140	33984	5040	0

Valoarea $E(n, k)$ este calculată în felul următor: luăm toate permutările secvenței $\{1, 1, 2, 2, ..., n, n\}$ ținând seama de faptul că între două apariții ale oricărui număr m nu există numere mai mici decât m.

În aceste condiții $E(n, k)$ este numărul permutărilor care conțin exact k subsecvențe crescătoare.

Să observăm că atunci când trecem de la $n - 1$ la n, cele două numere n pot fi inserate doar unul după altul. La fel ca în cazul triunghiului de speța I al lui *Euler*, și aici avem două cazuri în deducerea din permutările a $n - 1$ elemente.

Primul caz este absolut identic cu primul caz al triunghiului de speța I al lui *Euler*.

Pe de altă parte, în al doilea caz al n-lea număr poate fi pus în $2 \cdot n - 1 - k$ poziții în loc de $n - k$ poziții.

Din cele de mai sus rezultă că

$$E(n, k) = (k + 1) \cdot E(n - 1, k) + (2 \cdot n - 1 - k) \cdot E(n - 1, k - 1).$$

Punem valoarea lui n în celula `A1`. Bazându-ne pe aceasta, scriem în celula `A2` formula :

```
=IF(ROW()<=$A$1+2,ROW()-2,"").
```

Extindem această formulă pe 15 linii (o formulă similară se va scrie și în prima linie.)

Astfel, pentru $k = 0$, avem $E(n, k) = 1$ (există o singură secvență monoton descrescătoare), deci celula `B2` conține formula `=IF(A2="","",1)` cu care umplem coloana `B`.

Spațiul de deasupra triunghiului trebuie să-l lăsăm liber; bineînțeles pe diagonală trebuie să avem `0` ($E(n, n)$), iar celelalte elemente pot fi calculate din linia precedentă cu ajutorul formulei de mai sus.

Astfel conținutul celulei `C3` care se va extinde este:

```
=IF(OR(COLUMN()>ROW(), B3=""),
    IF(COLUMN()=ROW(),0,(COLUMN()-1)*
                C2+(2*ROW()-1-COLUMN())* B2)).
```

Concluzii

Soluțiile problemelor de mai sus și introducerea lor în concursurile de matematică organizate pentru elevi demonstrează faptul că problemele de combinatorică bazate pe umplerea tabelelor constituie o provocare la care elevii pot răspunde pozitiv.

Astfel se acordă șansa de a rezolva probleme de combinatorică cu metode netradiționale adică cu instrumente inventate pentru altfel de scopuri.

În scopul ilustrării afirmației de mai sus, vom încerca să rezolvăm o problemă de programare propusă spre rezolvare în cadrul *Concursului de Informatică pentru elevii de liceu* din *Ungaria* [2].

O secvență de caractere este un palindrom dacă aceasta este simetrică, adică, citind-o de la stânga la dreapta și de la dreapta la stânga, obținem aceeași secvență.

Un șir de caractere oarecare se poate transforma într-un palindrom, inserând în el anumite caractere.

De exemplu, dacă inserăm două caractere în secvența `S = 'Ab3bd'`, acesta se transformă într-un palindrom (putem obține `'dAb3bAd'` sau `'Adb3bdA'`). Inserând mai puțin de două caractere, nu putem transforma șirul de caractere într-un palindrom.

Determinați numărul minim de caractere prin inserarea cărora cuvântul S se transformă în palindrom.

Pentru orice pereche de indici (i, j), $1 \le i \le j \le n$, vom nota cu $M(i, j)$ numărul minim de caractere care trebuie inserate în cuvântul $S[i .. j] = S[i] .. S[j]$ pentru ca acesta să devină palindrom.

Soluția problemei va fi dată de $M(1, n)$.

Avem o problemă care se rezolvă folosind metoda programării dinamice, soluția putând fi descrisă astfel:

- $M(i, j) = 0$, dacă $i \ge j$;
- $M(i, j) = M(i + 1, j - 1)$, dacă $i < j$ și $S[i] = S[j]$;
- $M(i, j) = 1 + \min(M(i + 1, j), M(i, j - 1))$, dacă $i < j$ și $S[i] \ne S[j]$.

Literele cuvântului dat le vom așeza în linia `A` începând cu a doua coloană.

Sub această linie vom completa o matrice, care pe diagonala principală și sub diagonală conține zerouri, deci în celula `B2` vom scrie formula cu trei alternative unde prima condiție este:

```
=IF(ROW()>=COLUMN(),0,
```

A doua ramură descrie cazul când cele două litere potrivite sunt identice.

Vom folosi funcția `INDEX`. Dacă sunt identice, vom lua valoarea aflată în celula vecină din stânga jos:

```
IF(INDEX($A$1:$Z$1,1,ROW())=
  INDEX($A$1:$Z$1,1,COLUMN()),A4;
```

A treia ramură este simplă: vom crește cu 1 conținutul minimului celulelor aflate la stânga și sub celula curentă, astfel sfârșitul formulei devine:

```
1+MIN(B3;A2))).
```

Formula construită o extindem în mod corespunzător în jos din celula `B2`, apoi valorile din coloana `B` le extindem în dreapta pe distanța corespunzătoare.

Bibliografie

1. Roald R. Graham, Donald E. Knuth, Oren Patashnik, *Concrete Mathematics*, Addison-Wesley Publishing Company, Inc., 1994.
2. **http://nemes.inf.elte.hu/.**

Traducerea: Clara Ionescu

Problema CALENDARULUI

Tiberiu Socaciu

Conform tradiției creștine, se consideră numărarea anilor începând cu anul nașterii lui Hristos[1]. Alternativ la această numărătoare a anilor s-au folosit ab urbe condita[2] (în lumea romană), respectiv "de la facerea lumii", rămas de la evrei, sistem folosit și în zilele noastre în calendarul iudaic[3].

Anii bisecți și nebisecți

Calendarul, așa cum îl știm noi astăzi, cu lunile care îl compun de dimensiuni alternante (30 și 31 zile) a rămas de la romani.

Este un calendar astronomic bazat, în principiu, pe fazele lunii și care are o lună februarie care din patru în patru ani are o zi în plus.

Adăugarea acestei zile de către romani se făcea prin dublarea zilei de *6 februarie*, de unde și denumirea de an bisect[4].

La o primă vedere, verificarea dacă un an este sau nu bisect se poate realiza prin verificarea faptului dacă respectivul an (ca număr intreg) este congruent cu 0 modulo 4, adică:

```
((An % 4) == 0)
```

Acesta este unul din lucrurile cele mai importante în momentul în care dorim să efectuăm calculele de calendar.

A existat anul 0?

Una dintre principalele probleme care s-au pus la corelarea datelor calendaristice anterioare anului de pornire a "erei creștine", a fost numerotarea anilor precedenți.

Decizia luată în evul mediu, ca alternativă la mai vechiul a.u.c. (de la înființarea *Romei*)[5] a fost că înaintea anului 1 d.Hr. a fost anul 1 î.Hr., deci nu a existat niciun an 0[6].

Calendarele iulian și gregorian

Deoarece exista o mică diferență între 365,25 (cât ar fi fost dimensiunea medie a anului conform calendarului iulian) și 365,2422 care este o valoare mai exactă a anului astronomic[7] se impunea o ajustare a calendarului.

Ajunseseră să se strângă aproximativ zece zile, fapt aparent inofensiv, dar hotărâtor pentru determinarea corectă a datei *Paștelui*[8].

Pentru o aproximare mai bună a perioadei de 365,2422 regula exactă pentru anul bisect în calendarul *Gregorian* este: *un an este bisect dacă este divizibil cu 4, exceptând cazurile când este dizibil cu 100 fără a fi divizibil cu 400.*

Urmând această regulă durata medie a anului calendaristic este de 365,2425 zile care este o aproximație destul de bună[9].

---

[1] *în [IST], vezi probleme legate de corelarea acestei nașteri cu evenimentele istorice și astronomice în paralel cu evangheliile*

[2] *de la întemeierea legendară a Romei (vezi legenda lui Romulus, Remus și a lupoaicei)*

[3] *Sisteme alternative de numărare a anilor care mai sunt în vigoare întâlnim la toate marile civilizații care au reușit să supraviețuiască până astăzi prin urmași (de exemplu, chinezii și calendarul chinezesc)*

[4] *adica bi-sextus (sextus în latinî înseamnă șase)*

[5] *era varonica ab urbe condita, propusă anterior pentru data de 21 aprilie 753 î.Hr; calculul a fost efectuat în sec. I î.Hr. de către Marcus-Terentius Varro și Titus-Pomponius Atticus, care au adăugat 244 de ani (perioada de guvernare a primilor regi romani) la cei 509 ani scurși de la întemeierea legendară a Romei în anul 753 î.Hr. (vezi [IST])*

[6] *reconsiderarea anilor dinaintea nașterii lui Iisus (ante Christum natum) s-a realizat în sec. al XVII-lea de către Denis Petavius, fiind acceptată în Europa în cursul secolului al XVIII-lea; Recalcularea lui Denis Petavius nu a luat însă în considerare anul 0; eroarea a fost îndreptată în anul 1740 de către Cassini, care a adăugat un an suplimentar anilor ante Christum natum (vezi [IST]); părerea personală a autoului este că absența anului zero este datorată și lipsei conceptului algebrico-aritmetic de zero până în secolul al XVII-lea sau al XVIII-lea.*

[7] *a fost pentru prima dată propusă de doctorul Napolitan Aloysius Lilius și a fost decretată de Papa Grigore al XIII-lea, (numele căruia l-a primit calendarul), pe 24 februarie 1582*

[8] *calcul efectuat așa cum a decis Conciliul de la Niceea*

[9] *vezi [WIKI-B]*

Adoptarea calendarului gregorian[10]

Când a început folosirea noului calendar, pentru a corecta eroarea acumulată în cele 13 secole care trecuseră de la *Conciliul din Niceea*, s-a trecut la ștergerea a 10 zile din calendarul solar.

Ultima zi a calendarului iulian a fost **4 octombrie 1582** și a fost urmată de prima zi a calendarului gregorian, **15 octombrie 1582**.

Totuși, datele cuprinse între **5 octombrie 1582** și **14 octombrie 1582** (inclusiv) sunt încă valide în aproape toate țările, deoarece cea mai mare parte a țărilor catolice nu au adoptat noul calendar la data specificată în bula papală, ci doar câteva luni sau chiar câțiva ani mai târziu, (ultima în 1587).

[10] *vezi [WIKI-G]*

Ziua *Anului Nou* fusese deja standardizată în toată *Europa Occidentală* pe 1 ianuarie încă de la sfârșitul secolului al XV-lea și începutul secolului al XVI-lea, inclusiv în țările care deveniseră protestante între timp, precum *Germania*, *Suedia* sau *Anglia*.

Totuși, deși în *Anglia* ziua de 1 ianuarie a fost numită ziua *Anului Nou*, schimbarea numărului anului s-a realizat pe **25 martie** - *Lady Day* (*Bunavestire*) până în 1752.

Prin urmare, se acceptă 1752 ca fiind anul în care au dispărut din calendar zece zile în luna septembrie: astfel, este demn de remarcat modul în care este afișată luna septembrie în calendarul anului 1752[11] folosind comanda `cal` de afișare a calendarului pe un sistem tip *Unix*[12]. Structura calendarului este ilustrată în casetă.

[11] *Am folosit programul cal rulat pe un sistem linux cu Fedora Core 5*

[12] *Am folosit programul cal rulat pe un sistem linux cu Fedora Core 5*

Calendarul anului 1752

```
[root@server ~]# cal 1752
                                1752

      January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2   3  4  5  6  7  8  9   1  2  3  4  5  6  7
 5  6  7  8  9 10 11  10 11 12 13 14 15 16   8  9 10 11 12 13 14
12 13 14 15 16 17 18  17 18 19 20 21 22 23  15 16 17 18 19 20 21
19 20 21 22 23 24 25  24 25 26 27 28 29 30  22 23 24 25 26 27 28
26 27 28 29 30 31     31                    29 30 31

       April                 May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2   1  2  3  4  5  6  7      1  2  3  4  5  6
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   7  8  9 10 11 12 13
10 11 12 13 14 15 16  15 16 17 18 19 20 21  14 15 16 17 18 19 20
17 18 19 20 21 22 23  22 23 24 25 26 27 28  21 22 23 24 25 26 27
24 25 26 27 28 29 30  29 30 31              28 29 30
31

       July                  August                September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2   1  2  3  4  5  6  7      1  2 14 15 16
 3  4  5  6  7  8  9   8  9 10 11 12 13 14  17 18 19 20 21 22 23
10 11 12 13 14 15 16  15 16 17 18 19 20 21  24 25 26 27 28 29 30
17 18 19 20 21 22 23  22 23 24 25 26 27 28
24 25 26 27 28 29 30  29 30 31
31

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7                1  2  3  4                1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

Implementarea calculului calendarului

Astfel, implementarea unei funcții care să determine ziua din săptămână pentru o dată calendaristică de forma (an, lună, zi), ar fi de forma[13]:

```
/* return the 0 based day number for any
   date from 1 Jan. 1 to 31 Dec. 9999; assumes
   the Gregorian reformation eliminates
   3 Sep. 1752 through 13 Sep. 1752; returns
   Thursday for all missing days. */
int day_in_week(int day, int month, int year)
{
  long temp;
  temp = (long)(year - 1) * 365 +
         leap_years_since_year_1(year - 1) +
                day_in_year(day, month, year);
  if (temp < FIRST_MISSING_DAY)
    return ((temp - 1 + SATURDAY) % 7);
  if (temp >= (FIRST_MISSING_DAY +
                        NUMBER_MISSING_DAYS))
    return (((temp - 1 + SATURDAY) -
                  NUMBER_MISSING_DAYS) % 7);
  return (THURSDAY);
}
```

Sunt definite următoarele constante[14] folosite la calcul:

```
#define THURSDAY 4 // for reformation
#define SATURDAY 6 // 1 Jan 1 was a Saturday
#define FIRST_MISSING_DAY 639799
                                /* 3 Sep 1752 */
#define NUMBER_MISSING_DAYS 11
                        /* 11 day correction */
```

Trebuie remarcat faptul că funcția returnează un număr întreg cuprins între 0 și 6 pentru ziua din săptămână[15].

De asemenea, funcția returnează *Joi*[16] pentru fiecare zi din intervalul **3-14 Septembrie 1752**, zile care nu există.

O modalitate de calcul mai rafinată pentru numărul anilor bisecți ar putea fi[17]:

```
/* number of centuries since 1700, not
   inclusive */
#define centuries_since_1700(yr) \
        ((yr) > 1700 ? (yr) / 100 - 17 : 0)

/* number of centuries since 1700 whose modulo
   of 400 is 0 */
#define quad_centuries_since_1700(yr) \
     ((yr) > 1600 ? ((yr) - 1600) / 400 : 0)

/* number of leap years between year 1 and
   this year, not inclusive */
#define leap_years_since_year_1(yr) \
    ((yr) / 4 - centuries_since_1700(yr) + \
          quad_centuries_since_1700(yr))
```

Calculul numărului zilelor scurse de la începutul anului ar putea fi[18]:

```
/* return the 1 based day number within the
   year */
int day_in_year(int day, int month, int year)
{
  int i, leap;
  leap = leap_year(year);
  for (i = 1; i < month; i++)
    day += days_in_month[leap][i];
  return (day);
}
```

Verificarea faptului dacă anul este bisect sau nu este rafinată astfel:

```
/* leap year -- account for gregorian
   reformation in 1752 */
#define leap_year(yr) \
    ((yr) <= 1752 ? !((yr) % 4) : \
         (!((yr) % 4) && ((yr) % 100)) || \
                          !((yr) % 400))
```

Numărul zilelor pentru lunile unui an este dat de un tablou bidimensional cu 2 · 12 valori[19]:

```
static int days_in_month[2][13] = {
  {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
                                   30, 31},
  {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31,
                                   30, 31},
};
```

Bibliografie

[CAL] http://www.opensource.apple.com/darwinsource/Current/misc_cmds-17/cal/cal.c;

[IST] Radu Cerghizan, *Istoria calendarului*, **http://www.geocities.com/athens/delphi/5647/divtc07.htm**;

[MAN] man pentru comanda cal;

[WIKI-B] **http://ro.wikipedia.org/wiki/An_bisect**, pagina *Wiki* despre anul bisect;

[WIKI-G] **http://ro.wikipedia.org/wiki/Calendarul_gregorian**, pagina *Wiki* a calendarului gregorian;

[WIKI-I] **http://ro.wikipedia.org/wiki/Calendarul_iulian**, pagina *Wiki* a calendarului iulian.

[13] *vezi [CAL] unde este sursa versiunii 1.10 a programului cal folosit anterior*

[14] *cu rolul de a ajusta calculul zilei din săptămână pentru data calendaristică respectivă*

[15] *vezi [WIKI-B]*

[16] *adică 4*

[17] *vezi [CAL]*

[18] *vezi [CAL]*

[19] *a treisprezecea valoare este pentru indicele 0 care apare în limbajul de programare C*

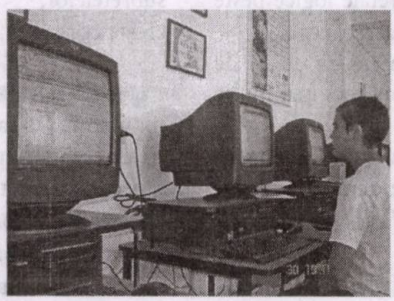
Școala VIITORULUI

Clasele viitorului în fiecare școală din România

Modernizarea educației tradiționale prin utilizarea eLearning-ului este o tendință naturală a evoluției și organizării lumii reale - tendința de virtualizare devenind un răspuns social la cererea crescândă de educație, la nevoia de formare continuă, pe multiple planuri: actorii implicați (formatori competenți, comunități mixte de învățare etc.), conținuturile vehiculate (suporturi și materiale didactice diverse), multitudinea de elemente didactice dezvoltate doar pe suport electronic, programe modulare, o varietate de conținuturi adiacente, complementare, proceduri de evaluare, proceduri de management instituțional, activități extrașcolare (resurse online, activități colaborative la distanță, participarea în comunități online de practică sau în campusuri virtuale etc.).

Datorită rolului de facilitare a accesului la informație și formare pe parcursul întregii vieți, calculatorul devine un instrument necesar oricărui cetățean al societății informaționale în care trăim. Accesul la un loc de muncă depinde din ce în ce mai mult de nivelul de competență demonstrat în lucrul cu calculatorul. Copiii care nu au acces la un calculator și nu știu să lucreze cu acesta vor fi marginalizați în societatea de mâine - societatea bazată pe cunoaștere.

Sistemele informatice, profesori și elevi reprezintă rețeta după care funcționează sistemul educațional în multe țări europene care au adoptat conceptul de Knowledge Economy (economia bazată pe cunoaștere). Cerințele tot mai ridicate și adaptarea la un principiu economic esențial "maximizarea rezultatelor și minimizarea eforturilor" a condus la transformarea și modernizarea sistemului educațional românesc. Pentru a învăța nu este suficientă o excursie la școală, ci implicarea activă a elevului și comunicarea permanentă profesor-elev. Școala viitorului trebuie să promoveze o educație de calitate și să fie axată pe valori în care elevii să creadă, în care să se regăsească,



îndeplinind două condiții fundamentale școala viitorului trebuie să le placă și să fie eficientă. *„Nu uitați că, până la urmă, calitatea educației depinde de atitudinea elevilor față de educație [...] precum și de expectanțele profesorilor în această privință."* (**Forumul Education Quality**)

Evoluția rapidă a sistemelor informatice a transmis și a cultivat o percepție pozitivă asupra procesului educațional, evidentă la fiecare copil care poate învăța altfel, mult mai inteligent - "***calculatorul te încurajează, te motivează și nu te ceartă***" este opinia elevilor care au experimentat "învățarea interactivă și creativă".

Calculatoare, eLearning și o școală a viitorului sunt trei noțiuni despre care putem vorbi și în România: peste 4800 de școli și licee au cel puțin un laborator informatizat și soft educațional care transformă total întregul proces de predare. Informatizarea sistemului educațional reprezintă un pas important ce face tranziția de la rutină, obligație, noțiuni greu de înțeles, memorare mecanică la motivație, curiozitate, emoția unei experiențe multimedia, o joacă inteligentă cu informațiile primite.

AeL, soluția de eLearning livrată prin intermediul programului SEI, este formată din platforma dedicată unei monitorizări și administrări eficiente a procesului educațional, și conținut educațional multimedia, instrument ce oferă elevilor plăcerea de a învăța interactiv orice lecție de matematică, fizică, chimie.

Suport eficient pentru administrarea resurselor din fiecare școală, platforma AeL a devenit asistentul fiecarui profesor și un instrument de comunicare eficient cu elevul, permițând totodată un feedback rapid de la fiecare elev și partciparea activă și permanentă la predarea noțiunilor. Comunicarea între profesor și elevi este facilitată de existența forumurilor unde pot fi deschise topic-uri variate referitoare la ora predată și la întrebările care au rămas fără răspuns.

Cu ajutorul lecțiilor AeL, elevii caută singuri și găsesc în câteva secunde informația dorită, esențială pentru înțelegerea unei probleme. Este foarte bine știut că procesul de învățare este un proces complex care implică atenția, percepția, memoria, imaginația, gândirea și motivația și este bine ca în predarea unei lecții să se utilizeze elemente multimedia - imagine, animație, culoare - elemente cu un efect psihologic pozitiv asupra fiecărui elev.

VLAN, TRUNK și STP

Tiberiu Socaciu

Domnul Sistemescu, prietenul nostru, are acum de lucru: învață. Are niscaiva probleme în rețea și a auzit că, dacă citește despre ARP-uri, își poate rezolva aceste probleme chiar dacă pe calculatoare sunt puse filtre de tip firewall. De asemenea, vrea să știe mai multe despre STP[1] pentru că a auzit că își poate face o mai bună gestiune a redundanței interconectărilor.

VLAN[2]-uri

VLAN-ul este o metodă de a construi rețele logice peste rețele fizice, dacă echipamentele permit acest lucru.

Mai exact, presupunem că avem un *switch* cu 24 de porturi, care ar trebui să deservească două sau mai multe rețele distincte din diverse motive[3].

Soluția pe care o avem la îndemână constă în realizarea unor *VLAN*-uri[4]: astfel, *VLAN*-urile se denumesc[5] folosind indicative numerice, fiecărui *VLAN* corespunzându-i un număr[6].

Astfel, la un moment dat, un port sau o interfață a echipamentului nu se poate afla în mai multe *VLAN*-uri simultan[7]. De asemenea, un *VLAN* poate avea mai multe porturi sau interfețe membre.

Avantajele utilizării *VLAN*-urilor sunt:

- mutarea unui calculator dintr-o rețea în alta se realizează prin schimbarea *VLAN*-ului atașat portului din *switch*;
- crește securitatea transmisiilor de date prin reducerea traficului de *broadcast*[8];
- se reduce necesarul de echipamente *hardware* care ar trebui folosite;
- ajută la managementul rețelelor și managementul creării subrețelelor;
- crește controlul asupra traficului în rețea.

Tag-area pachetelor

Comunicarea în cadrul echipamentelor care știu *VLAN*-uri se realizează prin pachete uzuale *Ethernet* modificate. Astfel, fiecărui pachet care trece prin mecanismul de *switching* i se atașează o informație suplimentară[9]: numărul *VLAN*-ului din care face parte, operația numindu-se ***tag-are***[10].

Care este utilitatea acestei *tag*-ări? Presupunem că avem două *switch*-uri, un *VLAN* 1 în primul *switch* și un *VLAN* 1 și în al doilea *switch*. Ce se întâmplă dacă interconectăm direct cele două *VLAN*-uri? În lipsa informațiilor de *tag*-are nu s-ar putea identifica rețele logice care se află pe un areal mai mare decât un singur echipament.

Să presupunem că suntem o firmă transportatoare și avem două *POP*-uri situate în locațiile *A* și *B*, *POP*-urile fiind interconectate (de exemplu, prin fibră optică). Avem un client care are două sedii care sunt legate la cele două *POP*-uri.

Acestui client îi putem atașa un același *VLAN* în ambele locații.

Switch-uri VLAN-transparent

După cum am spus, pachetele *Ethernet* care se "plimbă" prin *VLAN*-uri sunt *tag*-ate. Ce se întâmplă dacă între două *switch*-uri care știu *VLAN*-uri se interconectează un *switch* "chior"?

Vom remarca faptul că *VLAN*-ul construit de noi pe cele două echipamente nu se comportă ca o singură "rețea logică".

---

[1] *Spanning Tree Protocol*

[2] *cunoscut și sub numele de vLAN sau Virtual LAN*

[3] *de exemplu, suntem un transportator de date și legăm în NOC-ul nostru capete de rețea pentru mai mulți clienți; în mod evident, rețelele acestor clienți nu trebuie să interfereze sub nicio formă; o variantă ar fi conectarea rețelelor respective în switch-uri diferite; un alt caz ar putea apărea în cadrul unei firme care are rețele departamentale sau în cazul în care se dorește realizarea în paralel a mai multor rețele cu funcționalitate diferită și niveluri de clasificare a datelor*

[4] *realizarea efectivă se face prin conectarea la echipamentul în cauză (switch, router etc.) via telnet, ssh, pagină web sau direct din consolă*

[5] *de regulă*

[6] *de cele mai multe ori, în mod default toate porturile / interfețele fac parte dintr-un VLAN implicit*

[7] *deoarece, în acest caz, cele două VLAN-uri aparținând aceluiași domeniu de broadcast ar fi același VLAN*

[8] *vezi articolul despre ARP*

[9] *informația se mai numește* **tag**

[10] *în engleză,* **tagging**

Solutia ar fi ca *switch*-ul intermediar să fie și el unul care "știe" *VLAN*-uri sau, în cel mai rău caz, un *switch VLAN-transparent*, adică să comute și pachetele *Ethernet tag*-ate de echipamentele cu care este interconectat.

Trunking

Să presupunem că suntem o firmă care a cablat un cartier și a conectat apartamentele la *switch*-uri care știu *VLAN*-uri. De asemenea, am legat aceste *switch*-uri între ele, construind o rețea metropolitană.

Datorită faptului că rețeaua noastră este stabilă și fără probleme, mai mulți furnizori doresc să ofere servicii peste rețeaua noastră (internet, telefonie, televiziune digitală etc.).

Noi, ca furnizori de transport metropolitan și buclă locală la client, alocăm fiecărui furnizor câte un *VLAN*, singura problemă pe care mai trebuie să o rezolve furnizorii este să se "înțepe" în rețeaua noastră undeva, într-un port alocat *VLAN*-ului lor[11].

Presupunând că avem zece furnizori de utilități pentru care oferim acces la bucla locală; rezultă că ar trebui să avem un număr de cel puțin zece interconectări între toate *switch*-urile de pe traseu, fapt aproape imposibil.

Pentru a rezolva această problemă se folosește *trunking*-ul. Astfel, în loc să avem zece interconectări avem doar una singură, porturile în cauză, folosite la interconectare, fiind declarate *TRUNK*.

Putem spune[12] că un port *TRUNK* se află în mai multe *VLAN*-uri simultan[13].

Ca o paralelă, putem considera un canal de *trunking* ca un loc în care se multiplexează și demultiplexează comunicația *VLAN*-urilor pe baza *tag*-urilor de *VLAN* din pachete.

[11] numărul înțepăturilor trebuie să fie minim 1 (vom vedea ulterior cum putem avea mai multe înțepături cu rolul de creștere a redunanței)
[12] prin abuz de limbaj
[13] acest lucru este evident imposibil (am arătat anterior)

Prin urmare, putem spune că porturile sunt de două tipuri: *VLAN* și *TRUNK*; de asemenea, putem preciza că un port care este de tip *VLAN* are atașat un singur *VLAN* din care face parte.

Sunt toate VLAN-urile la fel?

Tot ce am spus până acum este la nivel principial. Lumea *VLAN*-urilor este dominată de protocolul de *tagging IEEE 802.1q*.

Până la impunerea acestui protocol, au existat mai multe protocoale de *tagging* cum ar fi protocolul *ISL*[14] sau protocolul VLT[15].

Trebuie remarcat faptul că *ISL* nu mai este suportat de mult de *Cisco* în sistemele sale de operare pentru *router*-e și *switch*-uri[16].

Practic, suntem interesați de tipul de *VLAN* folosit de echipamentul nostru în momentul în care suntem nevoiți să interconectăm două echipamente diferite și să ne asigurăm ca pachetele *tag*-ate trimise de primul echipament sunt recunoscute de cel de-al doilea. Din acest motiv, modul de *tag*-are este cel care definește practic sistemul de *VLAN/TRUNK*.

VLAN-uri 802.1q sub Linux

Interfața de lucru cu *VLAN*-uri în *Linux* este comanda `vconfig`, a cărei sintaxa de folosire este prezentată în caseta de pe această pagină[17].

Astfel, pentru a crea pe interfața fizică `eth0` *VLAN*-urile 2, 3 și 4, trebuie date comenzile:

```
# vconfig add eth0 2
# vconfig add eth0 3
# vconfig add eth0 4
```

[14] Inter-Switch Link, introdus de Cisco, ca o variantă a IEEE 802.10
[15] Virtual LAN Trunk, introdus de 3COM
[16] IOS și CATOS
[17] comanda a fost rulată pe un router Cisco Linksys cu sistem de operare Linux OpenWRT

Comanda vconfig

```
root@cuza-voda-8:~# vconfig
BusyBox v1.00 (2005.09.14-15:55+0000) multi-call binary

Usage: vconfig COMMAND [OPTIONS] ...

vconfig lets you create and remove virtual ethernet devices.

Options:
        add             [interface-name] [vlan_id]
        rem             [vlan-name]
        set_flag        [interface-name] [flag-num]         [0 | 1]
        set_egress_map  [vlan-name]      [skb_priority]     [vlan_qos]
        set_ingress_map [vlan-name]      [skb_priority]     [vlan_qos]
        set_name_type   [name-type]
```

Exemplu pentru VLAN-uri 802.1q sub Cisco

```
Cisco-Ariesului#vlan database
Cisco-Ariesului(vlan)#vtp transparent
Cisco-Ariesului(vlan)#vlan 2
Cisco-Ariesului(vlan)#vlan 3
Cisco-Ariesului(vlan)#exit
Cisco-Ariesului#configure terminal
Cisco-Ariesului(config)#int fastEthernet 0/1
Cisco-Ariesului(config-if)#switchport mode trunk
Cisco-Ariesului(config-if)#switchport trunk encapsulation dot1q
Cisco-Ariesului(config-if)#switchport trunk allowed vlan all
Cisco-Ariesului(config-if)#exit
Cisco-Ariesului(config)#int fastEthernet 0/2
Cisco-Ariesului(config-if)#switchport access vlan 2
Cisco-Ariesului(config-if)#spanning-tree portfast
Cisco-Ariesului(config-if)#exit
Cisco-Ariesului(config)#int fastEthernet 0/3
Cisco-Ariesului(config-if)#switchport access vlan 2
Cisco-Ariesului(config-if)#spanning-tree portfast
Cisco-Ariesului(config-if)#exit
```

De remarcat este faptul că după executarea acestor comenzi, au fost create interfețele `eth0.2`, `eth0.3` și `eth0.4`, pentru care se pot realiza deja setările aferente:

```
# ifconfig eth0.2 192.168.20.1 netmask 255.255.255.0
# ifconfig eth0.3 192.168.30.1 netmask 255.255.255.0
# ifconfig eth0.4 192.168.40.1 netmask 255.255.255.0
```

Aceste interfețe pot fi accesate până la ștergerea lor, care se realizează cu ajutorul comenzilor:

```
# vconfig rem eth0.2
# vconfig rem eth0.3
# vconfig rem eth0.4
```

VLAN-uri 802.1q sub Cisco

În caseta de mai sus este prezentată modalitatea de a seta pentru un *switch Cisco* portul 1 ca *TRUNK*, portul 2 ca *VLAN* 2, iar portul 3 ca *VLAN* 3[18].

[18] *exemplul este preluat din [CISCO] și modificat*

Comanda brctl

```
root@cuza-voda-8:~# brctl
Usage: brctl [commands]
commands:
        addbr           <bridge>                add bridge
        delbr           <bridge>                delete bridge
        addif           <bridge> <device>       add interface to bridge
        delif           <bridge> <device>       delete interface from bridge
        setageing       <bridge> <time>         set ageing time
        setbridgeprio   <bridge> <prio>         set bridge priority
        setfd           <bridge> <time>         set bridge forward delay
        sethello        <bridge> <time>         set hello time
        setmaxage       <bridge> <time>         set max message age
        setpathcost     <bridge> <port> <cost>  set path cost
        setportprio     <bridge> <port> <prio>  set port priority
        show                                    show a list of bridges
        showmacs        <bridge>                show a list of mac addrs
        showstp         <bridge>                show bridge stp info
        stp             <bridge> {on|off}       turn stp on/off
```

STP pentru bridge-uri

```
root@cuza-voda-8:~# brctl addbr br0
root@cuza-voda-8:~# brctl addif br0 vlan1
root@cuza-voda-8:~# brctl addif br0 eth1
root@cuza-voda-8:~# brctl stp  br0 on
root@cuza-voda-8:~# brctl showstp br0
br0
 bridge id               8000.000000971028
 designated root         8000.000000971028
 root port               0
 max age                 20.00
 hello time              2.00
 forward delay           15.0
 ageing time             300.00
 hello timer             0.00
 topology change timer   0.00
 flags

vlan1 (1)
 port id                 8001
 designated root         8000.000000971028
 designated bridge       8000.000000971028
 designated port         8001
 designated cost         0
 flags

eth1 (2)
 port id                 8002
 designated root         8000.000000971028
 designated bridge       8000.000000971028
 designated por          8002
 designated cost         0
 flags

 path cost               0
 bridge max age          20.00
 bridge hello time       2.00
 bridge forward delay    15.00

 tcn timer               0.00
 gc timer                0.00

 state                   disabled
 path cost               100
 message age timer       0.00
 forward delay timer     0.00
 hold timer              0.00

 state                   disabled
 path cost               100
 message age timer       0.00
 forward delay timer     0.00
 hold timer              0.00
```

Protocolul STP

Algoritmul implementat în *STP*[19] este cel de determinare a *arborelui parțial minim*[20].

Frumusețea implementării acestui algoritm rezidă în procesarea distribuită de către fiecare echipament în parte a informațiilor de topologie, care sunt primite prin mesaje de la *peer*-uri[21].

Încercați să executați o comandă `tcpdump` de urmărire a pachetelor într-un *switch* mai evoluat și veți observa la intervale regulate pachete de prezentare a echipamentului[22] către *peer*-uri.

Deși protocolul *STP* nu este legat explicit sau implicit de *VLAN*-uri[23], majoritatea echipamentelor care au una dintre facilități o are și pe cealaltă.

[19] *de către Radia Perlman de la DEC*

[20] *obținerea unui arbore dintr-un graf prin eliminarea succesivă a muchiilor de cost mare, fără a pierde din conexitate; astfel, la final rămânem cu un arbore care are doar muchii cu cost mic*

[21] *informații care au structura bine definită de protocolul IEEE 802.1d*

[22] *în protocolul 802.1d*

[23] *fiind vorba de două standarde diferite (802.1q și 802.1d).*

STP pentru interconectare redundantă

După cum probabil v-ați dat seama, *STP* este util pentru a crea redundanță. Astfel, dacă avem între două locații aflate în același *VLAN* o conexiune radio (de slabă calitate) și o conexiune pe fibra optică furnizată de un terț, vom pune un cost mare pe legătura prin radio și un cost mai mic pe legătura prin fibră optică.

În acest caz, în condiții normale, va fi preferat de cele două echipamente legătura cu cost mai mic, cealaltă legătură fiind folosită ca *back-up*.

Bridge-urile ca echipamente 802.1d

Reamintim din episodul anterior interfața de lucru cu *bridge*-uri (comanda `brctl`, prezentată din nou în caseta de pe pagina anterioară[24]).

Observăm că fiecare interfață de tip *bridge* se comportă ca un *switch* care, pentru fiecare din "porturile" sale[25], poate avea activat *STP*.

[24] *comanda este rulată pe același Linux OpenWRT*

[25] *mai exact, pentru fiecare interfață componentă a bridge-ului*

Comenzile care trebuie executate (împreună cu efectul obținut), sunt prezentate în cadrul casetei de pe pagina anterioară.

Soluțiile problemelor din episodul anterior

Problema #1

Demonstrați că relația "interfețele sunt în *bridge*" este o relație de echivalență.

Soluție

Pentru a demonstra echivalența este suficient să demonstrăm reflexivitatea, simetria și tranzitivitatea relației. Pentru demonstrarea proprietăților, este necesar să enunțăm următoarea lemă, a cărei demonstrație este imediată, pe baza celor prezentate în episodul anterior.

Lemă

Interfețele *A* și *B* sunt în *bridge* dacă și numai dacă orice pachet de tip *broadcast* de pe interfața *A* este transmis pe interfața *B* și invers.

- *reflexivitatea*: dacă un pachet *broadcast* este emis pe interfața *A*, el se va regăsi în mod evident pe interfața *A*, deci *A* și *A* sunt în *bridge*;
- *simetria*: presupunem că *A* și *B* sunt în *bridge*; pe baza lemei anterioare, orice pachet de *broadcast* de pe *A* se va regăsi și pe *B* și invers; așadar, tot pe baza acestei leme, *B* și *A* sunt în *bridge*;
- **tranzitivitatea**: presupunem că *A* și *B* sunt în *bridge*, respectiv *B* și *C* sunt în *bridge*; prin urmare, pe baza lemei anterioare (aplicată de două ori), un pachet de broadcast de pe *A* se va regăsi și în *B*, de unde va fi retransmis și în *C*; analog, un pachet de *broadcast* emis în *C* se va regăsi în final și în *A*; prin urmare, deducem că *A* și *C* sunt în *bridge*.

Problema #2

Avem la dispoziție trei rețele (*A*, *B*, *C*) și trei tuneluri (*A* - *B*, *A* - *C*, *B* - *C*).

Ce se va întâmpla dacă vom punem aceste trei tuneluri în *bridge*?

Noi ne propunem să obținem o oarecare redundanță. Este posibil?

Soluție

Trebuie activat *STL (Spanning Tree Protocol)*.

Problema #3

Aveți de proiectat un *router* cu interfață *wireless* încorporată.

Deoarece este instalat *Linux* pe acest sistem în *firmware*, v-ați gândit să puneți în *bridge* interfața *RF* cu interfața *TP*.

Este o idee bună sau nu? Argumentați.

Soluție

Deși majoritatea echipamentelor *WiFi* de pe piață realizează acest lucru pentru ca un client conectat *wireless* să se comporte identic cu unul conectat *wired*, noi nu recomandăm folosirea acestui mod de setare în special în cazul în care partea *TP* este conectată la un tronson de rețea cu multe pachete *broadcast* (de exemplu în rețeaua *MAN* a unui cartierist).

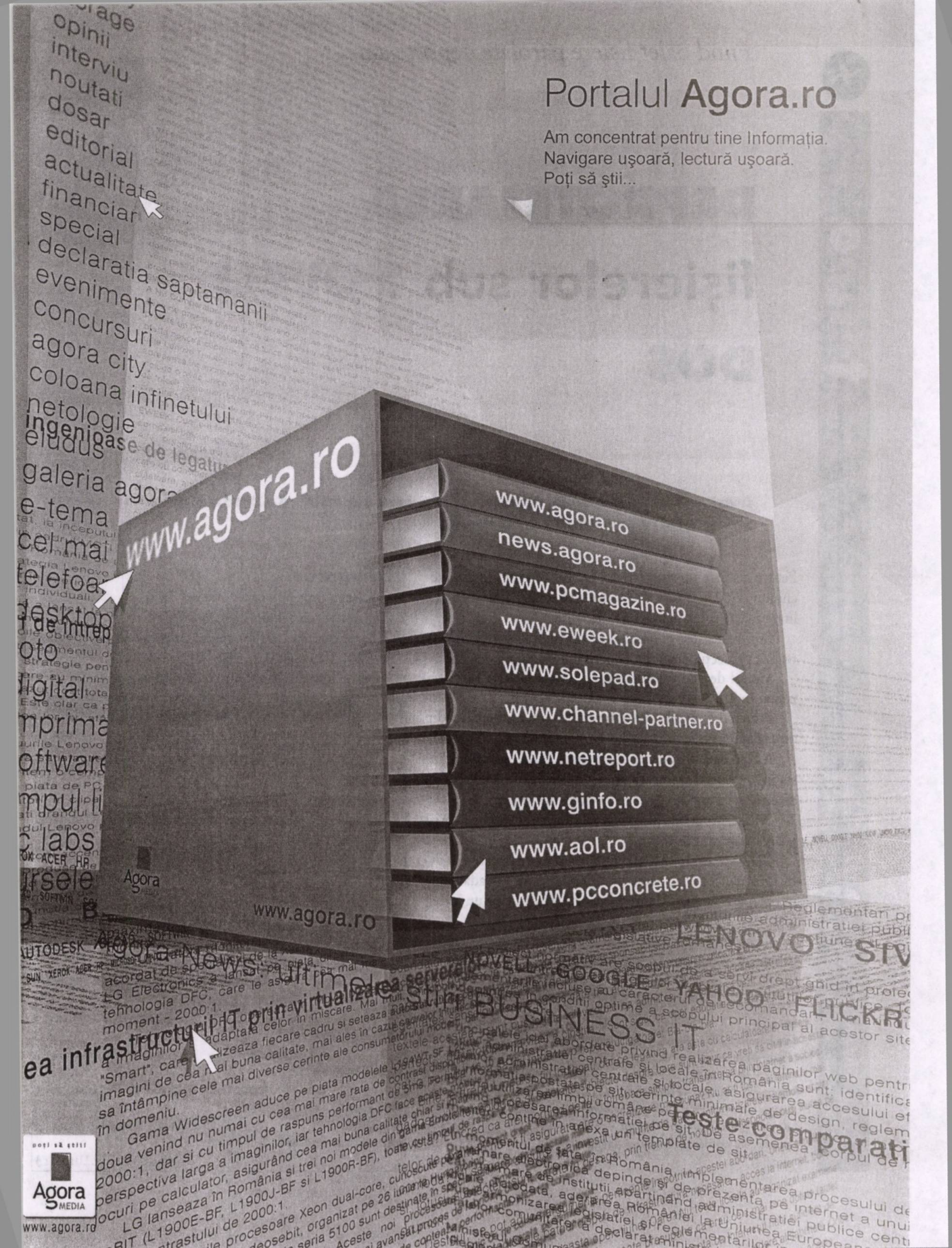
Exerciții și probleme

4. Furnizorul nostru de internet ne oferă la sediul central pe o fibră atât internet, cât și o legatură privată cu celelalte sedii. Cum poate face el acest lucru și ce avem noi de făcut pentru a rupe comunicația internet de comunicația privată?
5. Explicați funcționarea sistemului *WDS* în rețelele radio *WiFi*.
6. Vă rugăm să vă imaginați un sistem de *shaping* transparent între două echipamente *switch* care au capabilități de *STP*.
7. Presupunând că aveți un *PC* cu *n* interfețe *Ethernet*, l-ați putea transforma într-un *switch* inteligent. Cum realizați acest lucru?
8. Aveți un echipament tip *switch* care suportă ca *firmware* variante cu *Linux*. Explicați ce se întâmplă când treceți un port în *VLAN*, respectiv în *TRUNK*.
9. Aveți la dispoziție un sistem de *shaping* a traficului cam instabil (un *PC* cu probleme *hardware*) și un *switch* care știe *802.1d*. Cum proiectați un sistem de *shaping* cu o stabilitate mai mare folosind echipamentele pe care le aveți la dispoziție?

Răspunsurile pentru aceste întrebări vor fi prezentate în cadrul episodului următor.

Bibliografie

1. Cisco, *Configuring InterVLAN Routing and ISL/802.1Q Trunking on Catalyst 2900XL/3500XL/2940/2950/2970 Series Switches Using an External Router*, Document ID: 14976
2. Tiberiu Socaciu, *Informatica privită... altfel sau 16 chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft, 2006;
3. Tiberiu Socaciu, *Tuneluri și bridge-uri*, *GInfo* 16/8, decembrie 2006, Editura Agora Media, Târgu Mureș;
4. Tiberiu Socaciu, *Limitări de viteză în rețele*, *GInfo* 16/6, octombrie 2006, Editura Agora Media, Târgu Mureș;
5. Tiberiu Socaciu, *Lecții de rețelistică*, vol. 1, în curs de apariție;
6. Tiberiu Socaciu, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura InfoData, 2006;
7. Tiberiu Socaciu, *O altfel de ... Informatică sau diverse chestiuni de matematică, backtracking, clustere de calculatoare, Linux, inteligență artificială și networking*, Editura InfoData, 2007.


Portalul Agora.ro
Am concentrat pentru tine Informația.
Navigare uşoară, lectură uşoară.
Poți să ştii...
www.agora.ro
www.agora.ro
news.agora.ro
www.pcmagazine.ro
www.eweek.ro
www.solepad.ro
www.channel-partner.ro
www.netreport.ro
www.ginfo.ro
www.aol.ro
www.pcconcrete.ro
Agora
www.agora.ro
Agora News: ultimele
Stiri BUSINESS IT
Teste comparati
poți să știi
Agora MEDIA
www.agora.ro

DEPROTEJAREA fișierelor sub NOVELL DOS

Tiberiu Socaciu

Acest articol va prezenta două programe care exemplifică utilizarea funcțiilor sistem pentru accesarea harddisk-ului.

Pentru cei care nu știu, *Novell DOS 7.0*[1] este urmașul vechiului *DR-DOS 6.0*[2] de la care a moștenit o serie de utilitare foarte folositoare.

Printre acestea se numără și utilitarul PASSWORD[3], care realiza protecția prin parolă la fișiere pe partiții *FAT16*[4].

Modul de deprotejare pentru aceste fișiere a fost prezentat în [3].

Principiul constă în suprascrierea anumitor părți din disc și anume a locului unde era memorată criptat parola.

Având dată metoda de deprotejare, toată problema consta în identificarea intrării aferente în director folosind un *diskeditor*[5] și modificarea respectivei informații[6].

Primul program realizează acest lucru și anume deprotejarea unui singur fișier.

Cel de-al doilea program parsează toate intrările în directorul rădăcină, procesează fișierele și intră în adâncime pe directoare pentru deprotejarea fișierelor de pe un întreg disc.

Considerăm prezentul articol o bună reamintire a funcțiilor sistem pentru acces la *harddisk*[7].

[1] *vezi [1]*

[2] *vezi [2]*

[3] *cu ajutorul căruia se poate proteja accesul la un fișier sau director (Read/Write/Delete)*

[4] *vechiul format DOS, moștenit și de actualele sisteme de operare; în mare parte, FAT32 este similar cu FAT16, deci este posibilă o studiere a modului de protecție a sistemelor Windows, începând cu Windows '95*

[5] *de exemplu, cele din seria Norton Utilities*

[6] *practic, era suficientă zeroizarea octeților 14h și 15h*

[7] *vezi [4]*

Bibliografie

1. http://en.wikipedia.org/wiki/Novell_DOS;
2. http://en.wikipedia.org/wiki/DR-DOS;
3. Tiberiu Socaciu, *Deprotejarea fișierelor sub DR-DOS*, în *if* nr. 2/1992 (13), p. 58;
4. Tiberiu Socaciu, *Programare de sistem*, manuscris.

Listing FISIER.PAS (I)

```
{deprotejeaza un singur fisier}
program drdos_si_novelldos_unpassword;

const sector = 1024;
            { marimea maxima a unui sector }
const max = 10;
                      {lungimea maxima a caii}

var drive: Char;
 { unitatea pe care se realiza deprotejarea }
var buffer: array[0 .. sector -1 ] of Byte;
                                  {zona tampon}
var cale: array[1 .. max] of string[11];
    { lista de subdirectoare pana la fisier }
var curent: Word;
                               {sectorul curent}
var nr: Word;
         { lungimea caii (in subdirectoare) }
var raport: Word;
              { raportul sectoare / cluster }
var last: Word;
                          {ultimul sector citit}
var cluster: Word;
                              {urmatorul cluster}
```

Listing FISIER.PAS (II)

```
var nume: string[8];
                        { numele fisierului }
var ext: string[3];
                      { extensia fisierului }
var nrfat: Word;
                    { numarul de copii FAT }
var lengthfat: Word;
                  { lungimea unei copii FAT }
var adrfat: Word;
                  { adresa primei copii FAT }
var sectorlength: Word;
          { lungimea unui sector in octeti }
var i,j: Integer;
                  { variabile de uz general }

{ citeste un numar de n sectoare }
procedure citeste_sectoare(primul: Word;
                           cate: Word); assembler;
asm
 mov al,drive
 sub al,'A'
 mov dx,primul
 mov cx,cate
 lea bx,buffer
 int 25h
 pop dx
end;

{ scrie un numar de n sectoare }
procedure scrie_sectoare(primul: Word;
                         cate: Word); assembler;
asm
 mov al,drive
 sub al,'A'
 mov dx,primul
 mov cx,cate
 lea bx,buffer
 int 26h
 pop dx
end;

{ minimmul a doua valori }
function min(a, b: Word): Word;
begin
 if a < b then
  min := a
 else
  min := b
end;

{ numarul intrarii corespunzatoare lui s din
  ultimul sector citit }
function f(s: string): Integer;
var k, m: Word;
var p: string;
begin
  f := -1;
  m := 0;
  while m < sector do
  begin
    p := '';
    for k := 0 to 11 do
      p := p + Upcase(Chr(buffer[m + k]));
    if s = p then
  begin
      f := m;
      m := sector
    end;
    m := m + 32
 end
end;
```

Listing FISIER.PAS (III)

```
Begin
  Write('Unitatea: ');
  Readln(drive);
  drive := Upcase(drive);
  Write('Lungimea caii (inclusiv numele' +
                                  de fisier): ');
  Readln(nr);
  Writeln('Dati cele ', nr,
                    ' elemente ale caii');
  for i := 1 to nr do
  begin
    Write('Nume: ');
    Readln(nume);
    Write('Extensie: ');
    Readln(ext);
    cale[i] := '';
    for j := 1 to 11 do
      cale[i] := cale[i] + ' ';
    for j := 1 to min(8, Length(nume)) do
      cale[i][j] := Upcase(nume[j]);
    for j := 1 to min(3,length(ext)) do
      cale[i][8+j] := upcase(ext[j])
  end;

  { citeste sectorul de incarcare si
    realizeaza initializarile)
  citeste_sectoare(0, 1);
  nrfat := buffer[$10];
  lengthfat := 256 * buffer[$17]
                                + buffer[$16];
  adrfat := 256 * buffer[$0f] + buffer[$0e];
  sectorlength := 256 * buffer[$0c] +
                                  buffer[$0b];
  curent := nrfat * lengthfat + adrfat;
  data := curent + (256  * buffer[$12] +
     buffer[$11]) div (sectorlength div 32);
  raport := buffer[$0d];
  for i := 1 to nr do
    { determina intrarea pentru a i-a parte
      din cale }
  begin
    repeat
      citeste_sectoare(curent, 1);
      j := f(cale[i]);
      Inc(curent)
    until j >= 0;
    last := curent - 1;
    cluster := 256 * buffer[j + $1b] +
                                buffer[j + $1a];
    curent := data + (cluster - 2) * raport
  end;
  { deprotejeaza fisierul }
  buffer[j+$14] := 0;
  buffer[j+$15] := 0;
  scrie_sectoare(last, 1)
End.
```

Listing DIRECTOR.PAS (I)

```
{ deprotejeaza toate fisierele de pe un disc }
program drdos_si_novelldos_unpassword;

{ constante }
const sector = 1024;
const max = 10000;

{ variabile }
var drive: Char;
var ok: Boolean;
var s: string;
var adr: array[1..max] of Word;
var buf: array[0..sector - 1] of Byte;
var ii, i, j, n, curent, raport, data,
                        sectorlength: Integer;

{ citeste un numar de n sectoare }
procedure citeste_sectoare(primul: Word;
                        cate: Word); assembler;
asm
 mov al,drive
 sub al,'A'
 mov dx,primul
 mov cx,cate
 lea bx,buffer
 int 25h
 pop dx
end;

{ scrie un numar de n sectoare }
procedure scrie_sectoare(primul: Word;
                        cate: Word); assembler;
asm
 mov al,drive
 sub al,'A'
 mov dx,primul
 mov cx,cate
 lea bx,buffer
 int 26h
 pop dx
end;

{ programul principal }
Begin
  Writeln('Deprotejator DR DOS & NOVELL DOS');
  Writeln('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~');
  if paramcount = 0 then
    begin
      Write('Unitatea: ');
      Readln(drive);
      drive := Upcase(drive)
    end
  else
    begin
      s := paramstr(1);
      drive := upcase(s[1])
    end;
  citeste(0, 1);
  sectorlength := 256 * buf[$0c] + buf[$0b];
  n := 1;
  adr[1] := buf[$10] * (256 * buf[$17] +
      buf[$16]) + 256 * buf[$0f] + buf[$0e];
  data := adr[1] + (256 * buf[$12] + buf[$11])
                  div (sectorlength div 32);
  raport := buf[$0d];
  repeat
    curent := adr[n];
    Dec(n);
    ok := False;
```

Listing DIRECTOR.PAS (I)

```
    repeat
      { citeste sectorul curent }
      citeste(curent,1);
      Writeln('Citesc sectorul ',curent);
      { porneste cu prima intrare }
      j := 0;
      { parcurgand pe rand toate          }
      { intrarile din director            }
      for i := 1 to sectorlength div 32 do
      begin
        { daca nu este fisier sters,      }
        { eticheta de volum, director,    }
        { . sau ..                        }
        if not (buf[j] in [0, $e5, $2e]) and
           not (buf[j+$0b] and $08=$08) then
        begin
          { afiseaza numele fisierului    }
          { procesat                      }
          write('Procesez ');
          s := '';
          for ii := 0 to 7 do
            s := s + Chr(buf[j + ii]);
          while s[length(s)] = ' ' do
            Dec(s[0]);
          s := s + '.';
          for ii := 0 to 2 do
            s := s + Chr(buf[j + 8 + ii]);
          while s[length(s)] = ' ' do
            Dec(s[0]);
          if s[Length(s)] = '.' then
            Dec(s[0]);
          Writeln(s);
          { daca este protejat }
          if buf[j + $14] + buf[j + $15]
                                  > 0 then
      begin
            { deprotejeaza }
            buf[j + $14] := 0;
            buf[j + $15] := 0;
            buf[j + $0b] := buf[j + $0b]
                                  and $f8;
            writeln('Deprotejat!')
          end;
          { daca este subdirector }
          if buf[j + $0b] and $10 = $10 then
          begin
            { se va parcurge in adancime }
            inc(n);
            adr[n] := data + (256 * buf[j +
                 $1b] + buf[j + $1a] - 2) *
                                  raport
        end
      end
      else
        { s-au terminat intrarile in      }
        { directorul curent?              }
        ok := ok or (buf[j]=0);
      { trece la urmatoarea intrare }
      Inc(j, 32)
    end;
    { daca se iese din sector, atunci se  }
    { scrie sectorul                      }
    scrie(curent,1);
    { si se avanseaza la urmatorul sector }
    Inc(curent)
  until ok
until n = 0
End.
```

Principiul CONTORULUI

Tiberiu Socaciu

Cu toții știm ce este un contor. Doar avem câte unul pentru măsurarea consumului de energie electrică, pentru gazele naturale, pentru apă etc. În cadrul acestui articol ne referim la contor și la generarea configurațiilor posibile pentru un contor generalizat.

Câteva probleme propuse spre rezolvare

Propunem spre rezolvare câteva probleme:

- Să se rezolve ecuația

$$\overline{abc} = a^3 + b^3 + c^3$$

unde a, b și c sunt numere întregi.

- Să se rezolve ecuația:

$$\overline{abc} = a! + b! + c!$$

unde a, b și c sunt numere întregi.

- Să se afișeze toate reprezentările pe un octet ale numerelor hexazecimale (scrise în sistemul de numerație cu baza 16), cuprinse între `00H` și `FFH`.

- Să se rezolve ecuația:

$$a_1! + a_2! + \ldots + a_n! = a_1^3 + a_2^3 + \ldots + a_n^3$$

unde $a_1, a_2, \ldots, a_n$ sunt numere întregi.

Ce este un contor?

Știm cu toții cum arată un contor: l-am văzut la instalațiile de măsurare a consumului de apă, de energie electrică, de gaz, sau numărul de kilometri parcurși de un automobil, sau - de ce nu? - la numărul de pagini listate la copiatoare, imprimante laser etc.

În principiu, un contor este o instalație care permite afișarea unei secvențe de simboluri consecutive (de regulă cifrele sistemului zecimal).

De exemplu, un contor care are șase poziții va putea afișa un milion de secvențe distincte, cuprinse între `000000` și `999999`.

Prin urmare, un contor zecimal cu n poziții va putea afișa un număr de 10^n secvențe distincte cuprinse între `00...0` și `99...9` (în ambele cazuri avem n simboluri `"0"`, respectiv `"9"`).

Dacă mergem cu generalizarea mai departe și presupunem că fiecare poziție poate fi un simbol al unei mulțimi M care conține p elemente, atunci acest contor va putea "afișa" un număr de p^n secvențe distincte "cuprinse" între $m_1m_1...m_1$ și $m_pm_p...m_p$ (în ambele cazuri avem n simboluri "m_1", respectiv "m_p", unde m_1 și m_p sunt primul, respectiv ultimul element al mulțimii M, ordinea fiind dată de logica de construire a mulțimii M).

Mergând și mai departe cu generalizarea, dacă trebuie generate secvențe $x_1x_2...x_n$, astfel încât fiecare x_i să aparțină unei mulțimi M_i care conține p_i elemente, atunci numărul total de secvențe este $p_1 \cdot p_2 \cdot \ldots \cdot p_n$.

Principiul contorului

După cum se poate observa din problemele enunțate anterior, cea mai simplă rezolvare este folosind un contor cu trei poziții la primele două probleme, respectiv cu două poziții la cea de-a treia problemă, singura problemă care apare fiind generarea tuturor configurațiilor pentru contor. Rezolvările problemelor ar trebui să respecte următorul șablon:

```
pentru a1 ← 0, 9 execută
  pentru a2 ← 0, 9 execută
    pentru a3 ← 0, 9 execută
      dacă a1a2a3 este o configurație bună atunci
        scrie a1a2a3
      sfârșit dacă
    sfârșit pentru
  sfârșit pentru
sfârșit pentru
```

Presupunând că avem un contor cu k poziții (k este cunoscut), atunci am putea generaliza soluția problemei astfel:

```
pentru a1 ← 0, 9 execută
  pentru a2 ← 0, 9 execută
    pentru a3 ← 0, 9 execută
      ...
        pentru ak ← 0, 9 execută
          dacă a1a2a3...ak este o configurație bună
                                              atunci
            scrie a1a2a3...ak
          sfârșit dacă
        sfârșit pentru
      ...
    sfârșit pentru
  sfârșit pentru
sfârșit pentru
```

Dacă în plus, considerăm că fiecare cifră aparține unei mulțimi M cu p elemente (în cazul cifrelor zecimale avem $p = 10$), atunci secvența de program poate fi generalizată mai departe astfel:

```
pentru i1 ← 1, p execută
  a1 ← al i1-lea element al mulțimii M
  pentru i2 ← 1, p execută
    a2 ← al i2-lea element al mulțimii M
    pentru i3 ← 1, p execută
      a3 ← al i3-lea element al mulțimii M
      ...
        pentru ik ← 1, p execută
          ak ← al ik-lea element al mulțimii M
          dacă a1a2a3...ak este o configurație bună
                                              atunci
            scrie a1a2a3...ak
          sfârșit dacă
        sfârșit pentru
      ...
    sfârșit pentru
  sfârșit pentru
sfârșit pentru
```

Dacă considerăm, în plus, că fiecare element aparține unei mulțimi M_i cu p_i elemente, atunci secvența poate fi generalizată mai departe astfel:

```
pentru i1 ← 1, p1 execută
  a1 ← al i1-lea element al mulțimii M1
  pentru i2 ← 1, p2 execută
    a2 ← al i2-lea element al mulțimii M2
    pentru i3 ← 1, p3 execută
      a3 ← al i3-lea element al mulțimii M3
      ...
        pentru ik ← 1, pk execută
          ak ← al ik-lea element al mulțimii Mk
          dacă a1a2a3...ak este o configurație bună
                                              atunci
            scrie a1a2a3...ak
          sfârșit dacă
        sfârșit pentru
      ...
    sfârșit pentru
  sfârșit pentru
sfârșit pentru
```

Un studiu mai atent asupra rezolvărilor propuse ne arată că soluțiile de mai sus funcționează numai în cazul în care este cunoscută la scrierea programului numărul de poziții din contor.

În cazul în care acest număr este necunoscut (de regulă, acesta este cunoscut la rulare, prin citire), trebuie regândit întregul algoritm, astfel încât contorul să se incrementeze în mod corect.

Presupunând că avem un contor cu n poziții (n cunoscut doar în momentul rulării), atunci am putea generaliza astfel:

```
pentru i ← 1, n execută
  a1 ← 0
sfârșit pentru
repetă
  k ← n
  cât timp k > 0 și ak = 9 execută
    ak ← 0
    k ← k - 1
  sfârșit cât timp
  dacă k > 0 atunci
    ak ← ak + 1;
    dacă a1a2...ak este o configurație bună atunci
      scrie a1a2...ak
    sfârșit dacă
  sfârșit dacă
până când k = 0
```

Este demn de remarcat faptul că algoritmul de mai sus pornește cu o secvență de forma `000...0` și își încheie execuția când se ajunge din nou la această secvență (testul `k = 0`).

Exerciții și probleme

- Scrieți algoritmul pentru principiul contorului pentru cazul în care fiecare cifră aparține unei mulțimi M cu p elemente (în cazul cifrelor zecimale $p = 10$).
- Scrieți algoritmul pentru principiul contorului pentru cazul în care a_k reprezintă numărul de ordine în mulțimea M a poziției cu indicele k din contor.
- Scrieți programe *Pascal* pentru rezolvarea problemelor de la începutul articolului.
- Demonstrați că algoritmul *backtracking* cu funcția de continuare identic adevărată este echivalent cu algoritmul contorului.

MODALITATEA de REPARTIZARE COMPUTERIZATĂ a elevilor în LICEE pe baza unei MEDII calculate

Tiberiu Socaciu

Se cunoaște problema repartizării elevilor în licee pe baza mediilor obținute la concurs sau alte forme de departajare. În continuare vom prezenta procedura de repartizare pe un caz concret, procedură care face parte dintr-un program care a rulat cu date reale timp de mai mulți ani.

Presupunem că lista opțiunilor este codificată într-o tabelă **LOCURI.DBF** cu următoarea structură:

```
OPTIUNE, C(1)
SEMNIFICATIE, C(50)
LOCURI, N(3)
```

Fiecare opțiune este codificată printr-un singur caracter (se poate extinde ușor această codificare pe două sau mai multe caractere).

Tabela care conține candidații și notele obținute (s-a considerat că notele sunt de la examenele de *Matematică*, *Limba și Literatura Română* și, eventual, *Limba și Literatura Maternă*) este **ELEVI.DBF** și are structura prezentată în continuare:

```
NUME, C(50)
ROM, N(5,2)
MAT, N(5,2)
MAG, N(5,2)
LIMBA, L
MED, N(5,2)
OPTIUNI, C(10)
INTRAT, C(1)
```

Câmpul MED este folosit la calculul mediei, iar câmpul INTRAT este completat cu codul opțiunii care a fost aleasă de program pentru candidat (vezi semnificația din tabela **LOCURI.DBF**).

În cazul în care candidatul este respins, se completează de către program șirul "*" ca și opțiune.

Lista ordonată a preferințelor candidatului este memorată în șirul de caractere OPTIUNI, câte un caracter pentru fiecare cod de opțiune ales.

În casete este prezentată procedura de repartizare a candidaților.

Menționăm faptul că programul din care face parte această procedură a fost folosit timp de aproximativ cinci ani la *Liceul Teoretic "Lucian Blaga"* din *Cluj-Napoca* la admiterile în liceu, respectiv la admiterile în liceu din sesiunile extraordinare cu locurile rămase vacante în municipiul *Cluj-Napoca* la toate liceele.

Listing LICEE.PRG (I)

```
* repartizarea candidatilor pe profiluri
PROCEDURE repartizari
USE elevi.dbf

nrel = RECCOUNT()
min = 10
l = 10
max = 70

IF max - min > nrel
  min = INT((80 - nrel) / 2) + 2
  max = 78 - INT((80 - nrel) / 2)
ENDIF

DEFINE WINDOW geam FROM l - 1, min TO l + 1, ;
  max IN SCREEN DOUBLE SHADOW COLOR SCHEME 1

ACTIVATE WINDOW geam

DO faracursor

USE locuri.dbf

nroptiuni = 0

SCAN
  nroptiuni = nroptiuni + 1
  optiunile[nroptiuni] = semnificatie
  codificarile[nroptiuni] = optiune
  locurile[nroptiuni] = locuri
  mediile[nroptiuni] = 11
ENDSCAN

USE elevi.dbf

DELETE FILE bula.idx

INDEX ON -med TO bula

nrel = RECCOUNT()

i = 0

SCAN

  ok = .F.
  REPLACE intrat WITH ' '

  IF med = 0
    REPLACE intrat WITH '!'
    ok = .T.
  ENDIF

  IF limba
    ok = (rom < 5).OR.(mat < 5).OR.(mag < 5)
  ELSE
    ok = (rom < 5).OR.(mat < 5)
  ENDIF
```

Listing LICEE.PRG (II)

```
  IF ok.AND.(intrat#'!')
    REPLACE INTRAT WITH '*'
  ENDIF

  oo = optiuni
  nn = LEN(TRIM(oo))
  k = 1

  DO WHILE(.NOT.ok).AND.(k <= nn)
    jj = 0
    FOR ii = 1 TO nroptiuni
      IF codificarile[ii] = SUBSTR(oo, k, 1)
        jj = ii
      ENDIF
    ENDFOR

    IF (locurile[jj]>0).OR.((locurile[jj]<=0);
                        .AND.(mediile[jj]=med))
      ok = .T.
      locurile[jj] = locurile[jj]-1
      mediile[jj] = med
      REPLACE intrat WITH codificarile[jj]
    ENDIF

    k = k + 1
  ENDDO

  IF .NOT.ok.AND.(intrat=' ') THEN
    REPLACE intrat WITH '*'
  ENDIF

  @ 0, int((i/nrel)*(max-min-1)) SAY "?"
  i = i + 1

ENDSCAN

j = 0

USE locuri.dbf

SCAN
  j=j+1
  REPLACE libere WITH locurile[j]
  REPLACE ultmed WITH mediile[j]
ENDSCAN

USE

DO asteapta WITH 0
DO cucursor

DEACTIVATE WINDOW geam
RELEASE WINDOW geam

RETURN
```

PC MAGAZINE ROMÂNIA

www.pcmagazine.ro

discountul nostru

abonamentul tău

Abonează-te

Noul an. Se întrec magazinele în a-ți oferi reduceri, discounturi, soldări, promoții, spune-le cum vrei.
N-am vrut să rămânem de căruță: abonează-te acum la **PC Magazine România** și beneficiezi de oferta noastră promoțională: dublăm automat perioada pe care te abonezi la PC Magazine, altfel spus, plătești o revistă dar primești două! Într-adevăr, asta înseamnă **50%** reducere! Am mărit bine cifrele, ca să fie cât mai clar...

Pentru a te abona caută în revistă talonul de abonament și expediază-l pe adresa PC Magazine.

50%

PLĂTEȘTI 1, PRIMEȘTI 2
2:1

poți să știi!
Agora MEDIA
www.agora.ro

Promoție valabilă până la 31 ianuarie 2007.

Cabinetul
Călin Tatomir

Noi ştim cum

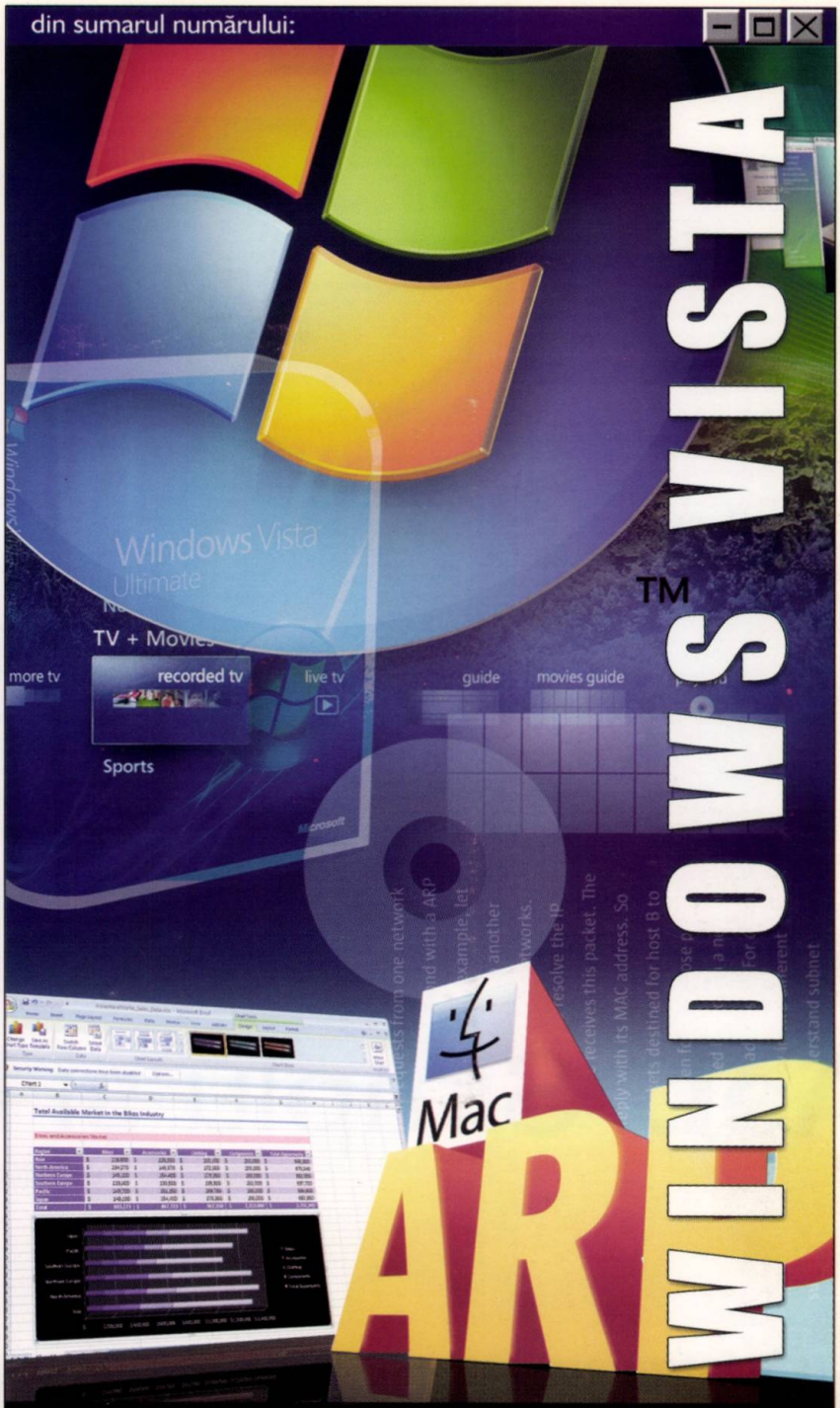
Gfo
Februarie 2007
Vol. 17/2
40.000 LEI / 4 LEI NOI

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

O publicație Agora Media

din sumarul numărului:



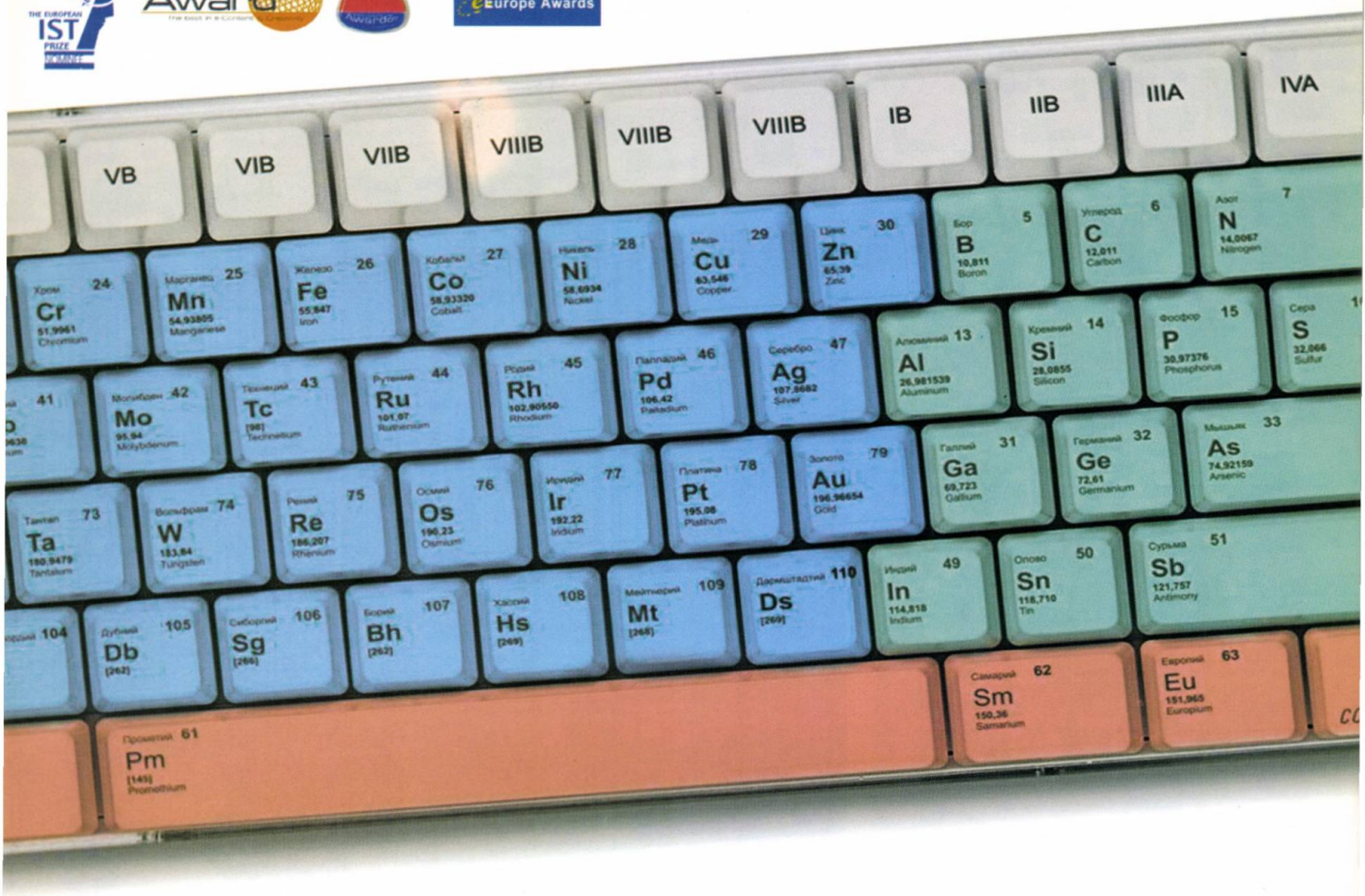
actual
Office 2007

info
UNIREA 2007

babel
POKER

serial
MAC și ARP

5 948373 000645 02



AeL. Simplu şi la obiect.

Lecţii interactive pentru noua generaţie.

Orice întrebare are un răspuns!

Noutăţi, suport tehnic, competiţii,
forum : www.portal.edu.ro

Orice problemă are o soluţie!

Centrul de suport tehnic de luni până
vineri între orele 8.00-20.00
Telverde : 0800410444 (linie gratuită)

Orice cauţi poate fi găsit!

Ai nevoie de o anumită lecţie electronică?
Întâi de toate, caut-o în biblioteca virtuală
AeL. Odată găsită, descarc-o gratuit!

AeL
Built to teach intelli-gently

sei

SIVECO
ROMANIA SA

Vol. 17, Nr. 2 /2007

gazeta de informatică

Σ Sumar

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tiberiu Socaciu

Adresa pentru corespondență:
Computer Libris Agora,
Șerpuitoare 25, Cluj-Napoca

Adresa e-mail: office@ginfo.ro

Tel: 0264/534.911, 0265/266.516
Fax: 0265/266.290

http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NET Report*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

O publicație Agora MEDIA

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu

Difuzare:
Laurențiu Nicolae
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

actual.. PAG. 5

Windows VISTA
Mihai SCORȚARU

Office 2007
Mihai SCORȚARU

info.. PAG. 9

UNIREA 2007

probleme.. PAG. 16

UNIREA 2007

e-learning.. PAG. 19

Șansa de a deveni CAMPION

babel.. PAG. 20

POKER
Mihai SCORȚARU

serial.. PAG. 40

MAC și ARP
Tiberiu SOCACIU

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

Singurele condiții pe care trebuie să le îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre și articolul să fie original (să nu mai fi fost publicat anterior și să nu fie o traducere a unui articol publicat de un alt autor).

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii.

Redacția GInfo nu își asumă nicio răspundere cu privire la informațiile prezentate în articole publicate.

Pentru articolele trimise la redacție, autorul cedează către GInfo dreptul exclusiv de publicare ale materialelor respective. Aceste articole nu pot fi publicate în alte publicații decât cu acordul scris al redacției sau în cazul în care redacția comunică autorului că articolul nu va fi publicat în revista GInfo.

Articolele publicate în revista GInfo nu pot fi preluate și publicate sub nicio formă decât cu acordul scris al redacției. Totuși, autorii articolelor pot publica integral sau parțial pe site-urile personale articolele cu GInfo, dar doar după 180 de zile de la publicare și cu menționarea faptului că articolul a fost publicat mai întâi în GInfo.

Pentru mai multe detalii vizitați:
http://www.ginfo.ro/revista/publicare.shtml

Cea mai bună introducere din lume în rețelele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine

EDIȚIA A PATRA

Rețele de calculatoare

ANDREW S. TANENBAUM

Cea mai bună introducere în arhitectura calculatoarelor - complet actualizată, reflectă ultimele schimbări din dinamica lume a calculatoarelor

Andrew S. Tanenbaum

ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR

Ediția a IV-a

SISTEME DE OPERARE MODERNE

EDIȚIA A DOUA

Andrew S. Tanenbaum

Această carte de referință, bazată pe exemple concrete, oferă informații practice de primă mână pentru înțelegerea și construirea sistemelor de operare moderne.

THOMAS H. CORMEN CHARLES E. LEISERSON RONALD R. RIVEST

INTRODUCERE ÎN ALGORITMI

Cartea tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, fiind binevenita elevilor, studenților și profesorilor

CĂRȚILE ESENȚIALE ▸

...cărțile care îți însoțesc cariera

Comandă-le pe toate și le iei cu 20% mai ieftin,
iar dacă ești student, cu 30%.

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

Un nou COMANDOR

Bill Gates a fost în România... El a inaugurat Centrul Global de Suport Tehnic al Microsoft, a lansat noul sistem de operare Windows Vista, precum și pachetul de aplicații Office 2007. Dar el s-a întâlnit și cu președintele României și cu primul ministru...

William Henry Gates III a fost decorat de către președintele *României* cu ordinul *Steaua României* în grad de comandor, adăugându-și astfel un nou titlu în "colecție".

Cel mai interesant titlu (deși este doar onorific) este acela de *Cavaler Comandor al Ordinului Imperiului Britanic* (*Knight Commander of the Order of the British Empire*), acordat de *Regina Elisabeta a II-a* a *Marii Britanii* în 2005. Acest titlu îi permite să își adauge la nume sufixul *KBE* și, dacă ar fi fost supus al reginei, prefixul *Sir*. Dar, cum *Bill Gates* este american, el nu poate fi numit *Sir William*.

Ordinul Imperiului Britanic are cinci clase (cavaler mare cruce, cavaler comandor, comandor, ofițer și membru), *Bill Gates* fiind "încadrat" la a doua ca importanță.

În schimb, *Steaua României* are șase grade (colan, mare cruce, mare ofițer, comandor, ofițer și cavaler), iar *Bill Gates* a fost încadrat la a patra categorie. El este "coleg" cu diferiți foști sau actuali ambasadori la *București*, politicieni străini sau oameni de cultură. Există și români care au acest grad; printre aceștia se numără gimnasta *Nadia Comăneci*.

Totuși, se pare că britanicii îl prețuiesc ceva mai mult decât noi pe miliardarul american.

Bill Gates nu a părut prea impresionat de titlul și atenția care i-au fost acordate. A ascultat discursul președintelui, a zâmbit atunci când președintele a sugerat că pirateria are și părțile ei bune (garantul *Constituției*?), a ascultat și discursul premierului și apoi a trecut la treabă.

Nu s-a implicat în chestiuni protocolare, nu a spus că știe despre *Dracula*, despre *Ceaușescu* sau alți români mai mult sau mai puțin celebri; și-a îndeplinit rolul și a pornit repede spre *Sofia*, fiindcă și bulgarii aveau nevoie de lansarea noului sistem de operare.

Totuși, trebuie să admitem faptul că momentul în care cel mai bogat om al planetei vizitează o țărișoară ca a noastră este unul destul de important.

Dar... *Bill Gates* a venit și a plecat (și este foarte posibil să nu mai

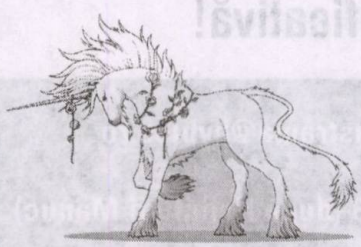
vină niciodată), iar noi avem un nou număr al revistei, care conține foarte puține articole, dar articolele sunt destul de ample.

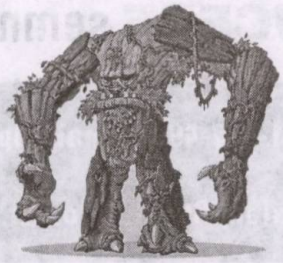
Am prezentat pe scurt *Vista* și *Office 2007*, ne-am ocupat de un concurs de *software* pentru elevi, programare și matematică, am continuat serialul dedicat rețelelor și am ocupat aproape jumătate din revistă cu un articol prin care încercăm să popularizăm limbajul de programare *Java* (nu jocul de *poker*, așa cum ar putea părea la prima vedere).

Este pentru prima dată când nu avem articole suficiente pentru a avea șapte niveluri de dificultate; de fapt, avem doar patru (spiriduș, vânător, druid și unicorn; se pare că dragonul de smarald nu a avut parte de articole cu nivel ridicat de dificultate). Dacă ar fi fost articole "pentru toată lumea", ar fi fost cele de pe această pagină (cu elfi; magicienii vor apărea în numărul următor).

Și, cum acesta ar putea fi ultimul număr înainte de olimpiade, aș dori să închei prin a ura succes tuturor celor care vor participa la concursurile din acest an.

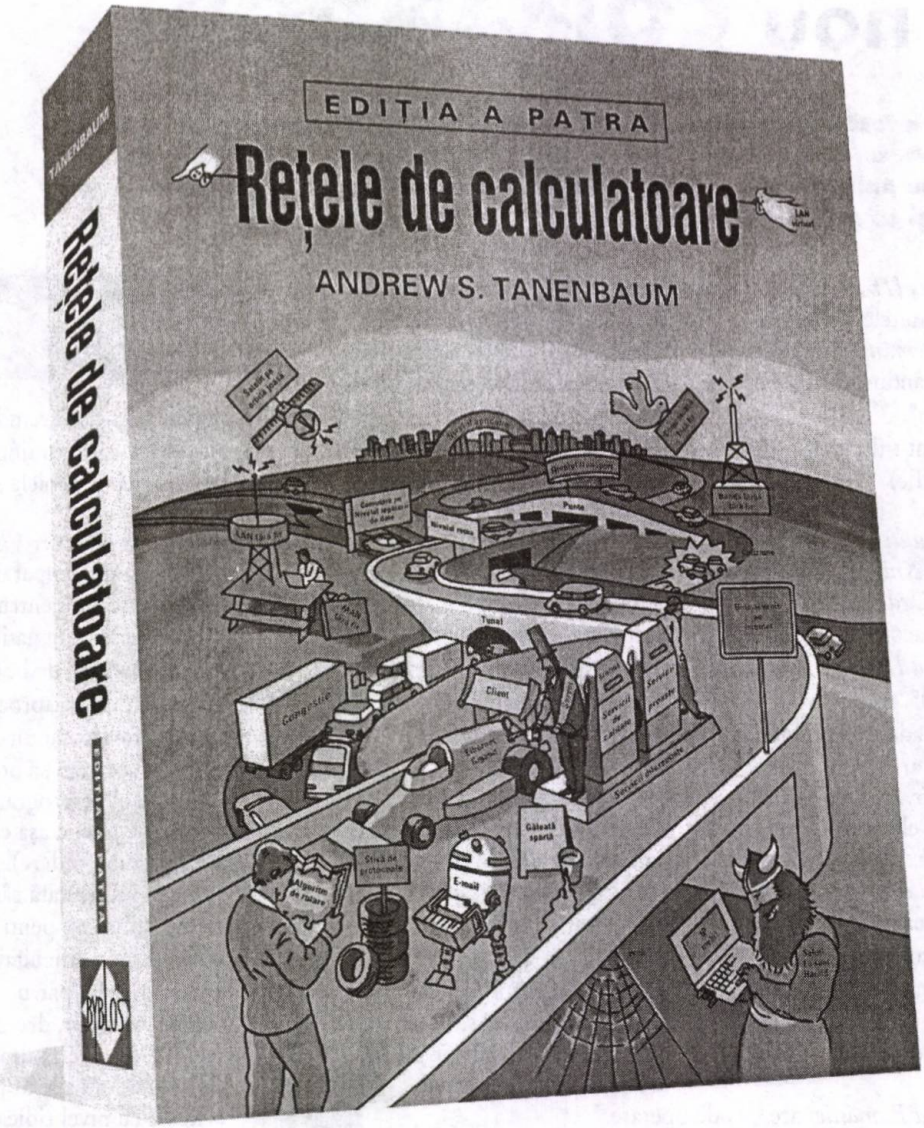
Mihai Scorțaru





Cine are carte, are parte!

...de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețelele de calculatoare

- complet actualizată pentru tehnologiile cheie de mâine -

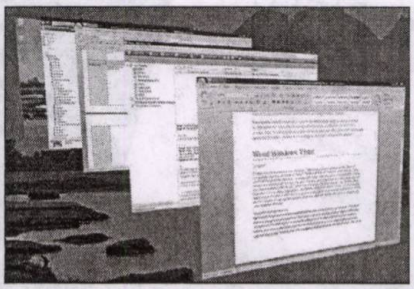
Comand-o acum și dacă ești student primești și o REDUCERE semnificativă!

Byblos srl, tel: 0318-056-314, 0721-927-499, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librărie: Galeria NOI-La Calu' Bălan, Str. Halelor Nr. 3 (după Hanul lui Manuc).

Windows VISTA

Mihai Scorțaru

Noul sistem de operare de la Microsoft a fost lansat, pentru publicul larg, pe data de 30 ianuarie 2007. În România, Windows Vista a fost lansat după două zile, în prezența fondatorului companiei Microsoft, Bill Gates.



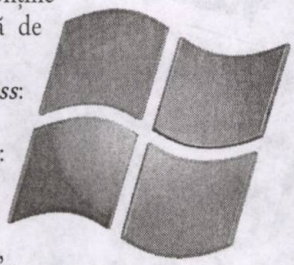
Dezvoltarea sistemului de operare *Windows Vista* s-a încheiat pe data de 8 noiembrie 2006; în următoarele două luni a fost distribuit abonaților *MSDN*, *TechNet Plus* și *TechNet Direct*. Pe data de 30 ianuarie 2007, sistemul de operare a devenit disponibil publicului larg.

Potrivit *Microsoft*, există sute de componente și facilități noi; cele mai importante dintre ele sunt:

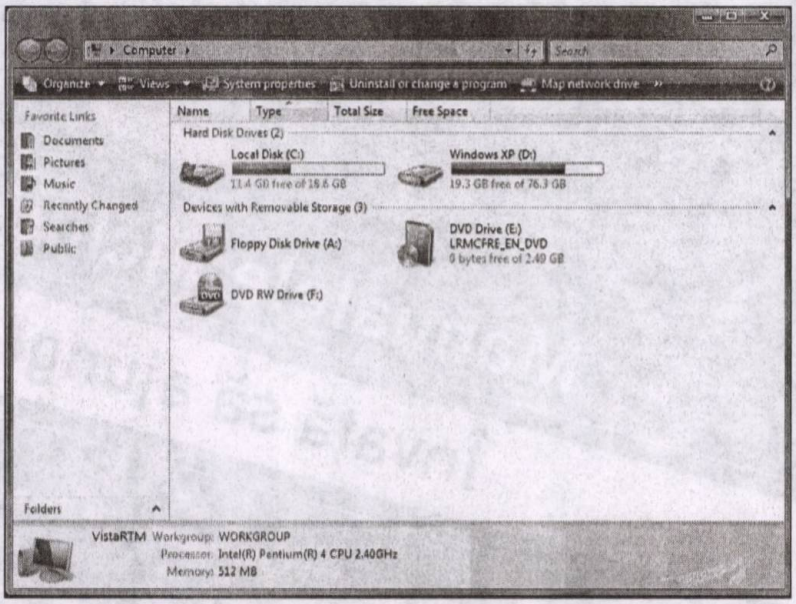
- ***Windows Aero***: o nouă interfață grafică;
- ***Windows Shell***: un nou *shell*, foarte diferit de cel din *Windows XP* (noul *Windows Explorer* este prezentat într-una dintre imagini);
- ***Windows Search***: noi facilități de căutare;
- ***Windows Sidebar***: un panou transparent, ancorat de marginea ecranului, în care se poat adăuga așa-numitele ***desktop gadgets***;
- ***Windows Internet Explorer 7***;
- ***Windows Media Player 11***.

Există mai multe variante ale sistemului de operare, destinate diferitelor categorii de utilizatori sau numai anumitor regiuni ale *Globului*:

- ***Windows Vista Starter***: pentru piețele emergente;
- ***Windows Vista Home Basic***: pentru utilizatorii cu bugete limitate, care nu au nevoie de suport media avansat;
- ***Windows Vista Home Premium***: pentru utilizatorii casnici - conține câteva facilități în plus față de varianta *Home Basic*;
- ***Windows Vista Bussiness***: pentru companiile mici;
- ***Windows Vista Enterprise***: pentru companiile mari;
- ***Windows Vista Ultimate***: versiunea cea mai avansată, care înglobează toate facilitățile oferite companiilor și utilizatorilor casnici.

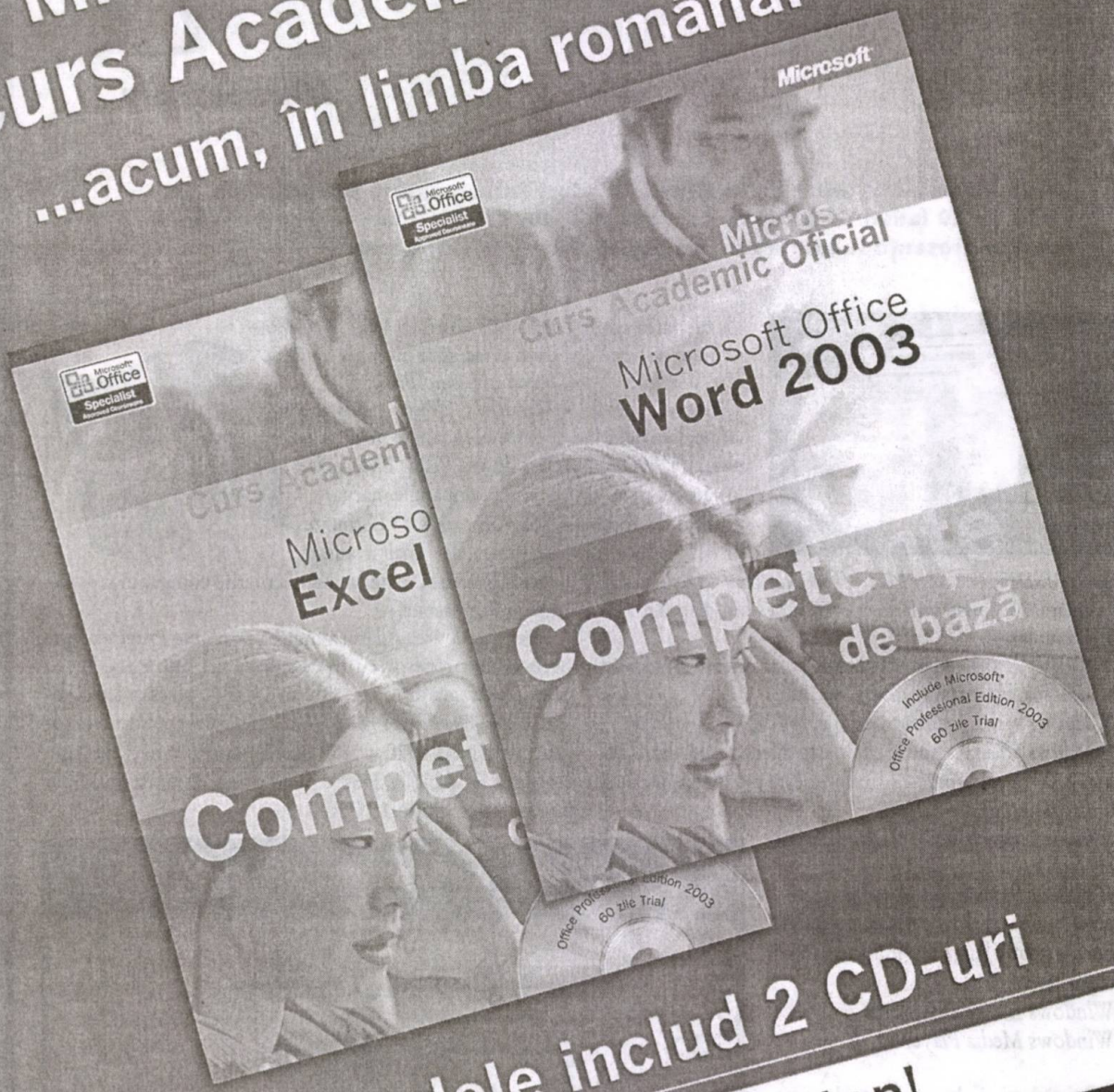


Pentru a putea rula varianta *Vista Home Premium* este necesar un procesor cu frecvența de cel puțin 1 GHz, o cantitate de memorie de cel puțin 1 GiB, o placă video compatibilă *DirectX* 9 cu 128 MiB de memorie (pentru o rezoluție de până la 1920 × 1200 pixeli) sau 512 MiB (pentru o rezoluție de până la 2560 × 1600 pixeli) și un spațiu liber pe disc de cel puțin 15 GiB.



NOU!

Microsoft®
Curs Academic Oficial
...acum, în limba română!



Manualele includ 2 CD-uri

Învață să ajungi în top!

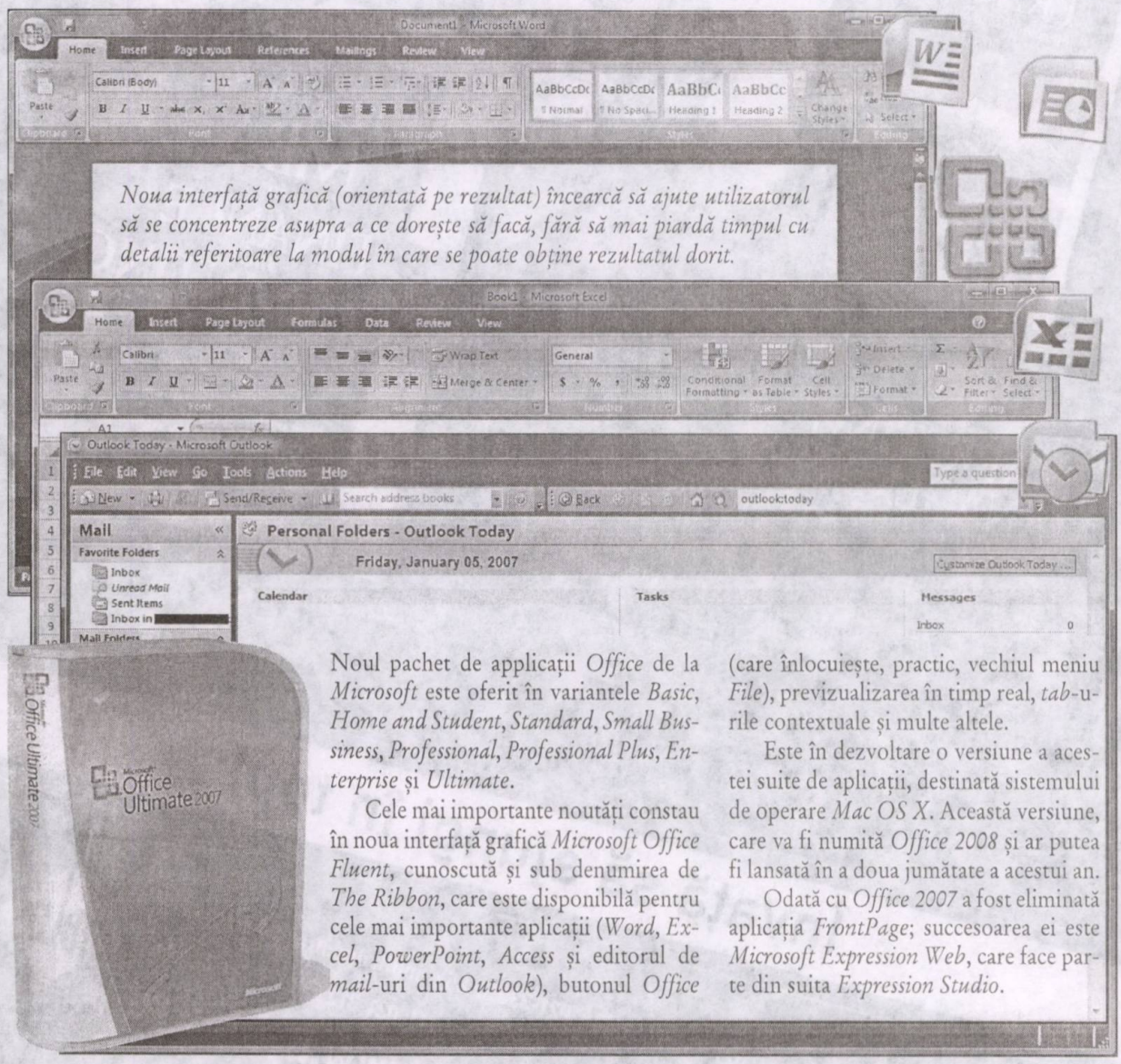
Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Office 2007

Mihai Scorțaru

Simultan cu noul sistem de operare Windows Vista, Microsoft a lansat o nouă versiune a suitei de aplicații pentru birou. Noul Office 2007 a fost lansat pe data de 30 noiembrie 2006 pentru clienții bussiness și pe 30 ianuarie 2007 pentru publicul larg.



Noua interfață grafică (orientată pe rezultat) încearcă să ajute utilizatorul să se concentreze asupra a ce dorește să facă, fără să mai piardă timpul cu detalii referitoare la modul în care se poate obține rezultatul dorit.

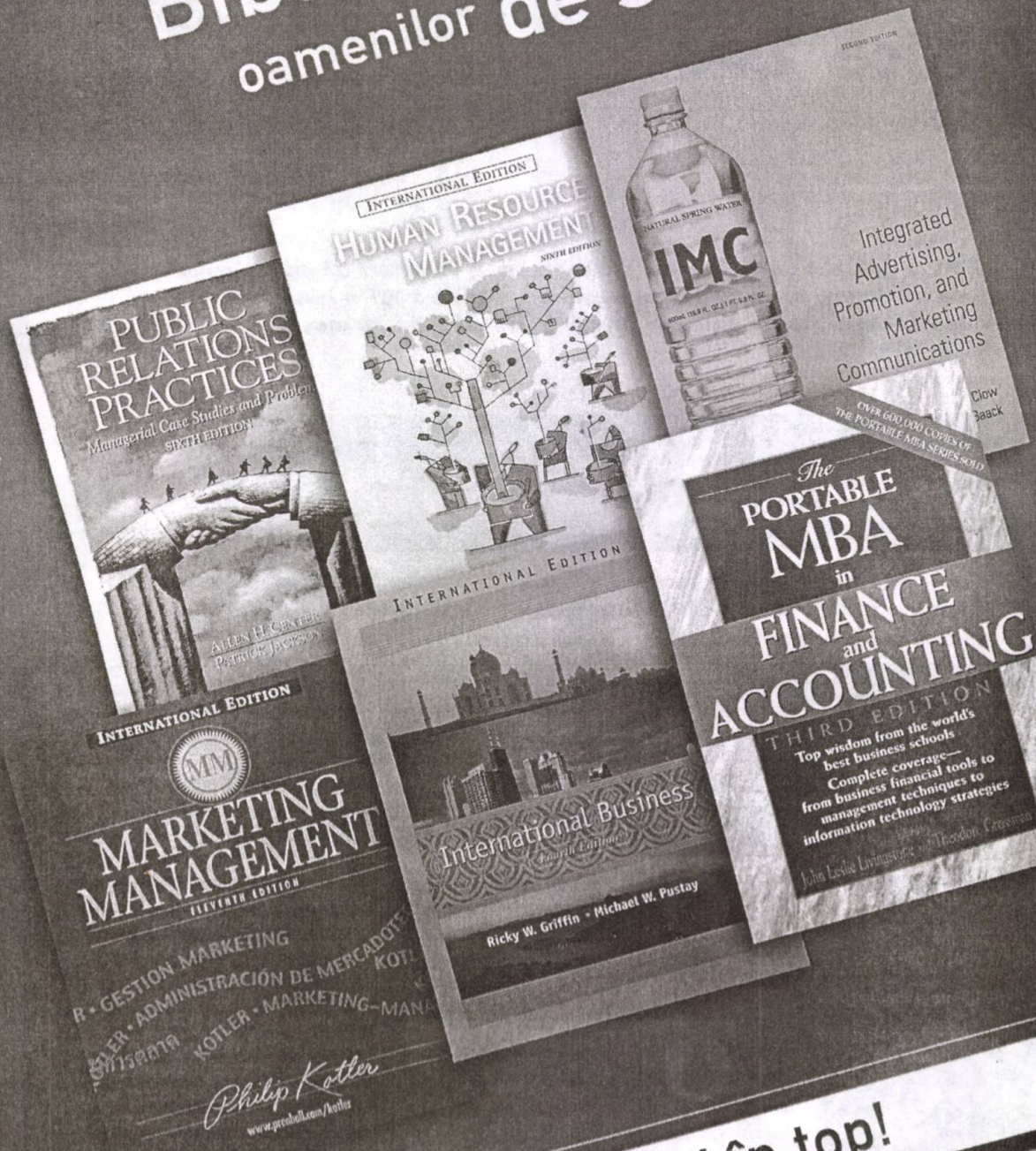
Noul pachet de aplicații *Office* de la *Microsoft* este oferit în variantele *Basic*, *Home and Student*, *Standard*, *Small Bussiness*, *Professional*, *Professional Plus*, *Enterprise* și *Ultimate*.

Cele mai importante noutăți constau în noua interfață grafică *Microsoft Office Fluent*, cunoscută și sub denumirea de *The Ribbon*, care este disponibilă pentru cele mai importante aplicații (*Word*, *Excel*, *PowerPoint*, *Access* și editorul de *mail*-uri din *Outlook*), butonul *Office* (care înlocuiește, practic, vechiul meniu *File*), previzualizarea în timp real, *tab*-urile contextuale și multe altele.

Este în dezvoltare o versiune a acestei suite de aplicații, destinată sistemului de operare *Mac OS X*. Această versiune, care va fi numită *Office 2008* și ar putea fi lansată în a doua jumătate a acestui an.

Odată cu *Office 2007* a fost eliminată aplicația *FrontPage*; succesoarea ei este *Microsoft Expression Web*, care face parte din suita *Expression Studio*.

Biblioteca oamenilor de succes



Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Focșani, jud. Vrancea

UNIREA 2007

În perioada 26 - 28 ianuarie, Colegiul Național "Unirea" din Focșani, primăria orașului, Inspectoratul Școlar Județean Vrancea și Microsoft România au organizat, în cadrul manifestărilor "Zilele Unirii", întâlnirea cu tema "Resurse IT&C în educație". Au fost organizate, printre altele, un concurs de proiecte, un concurs de probleme de informatică și unul de matematică.

Despre concurs

Concursul de *software* și pagini *Web* dedicat elevilor este foarte asemănător primei faze a concursului *InfoEducație*. Au existat trei secțiuni ale concursului și anume *software* educațional și utilitar, pagini *Web* și *multimedia*.

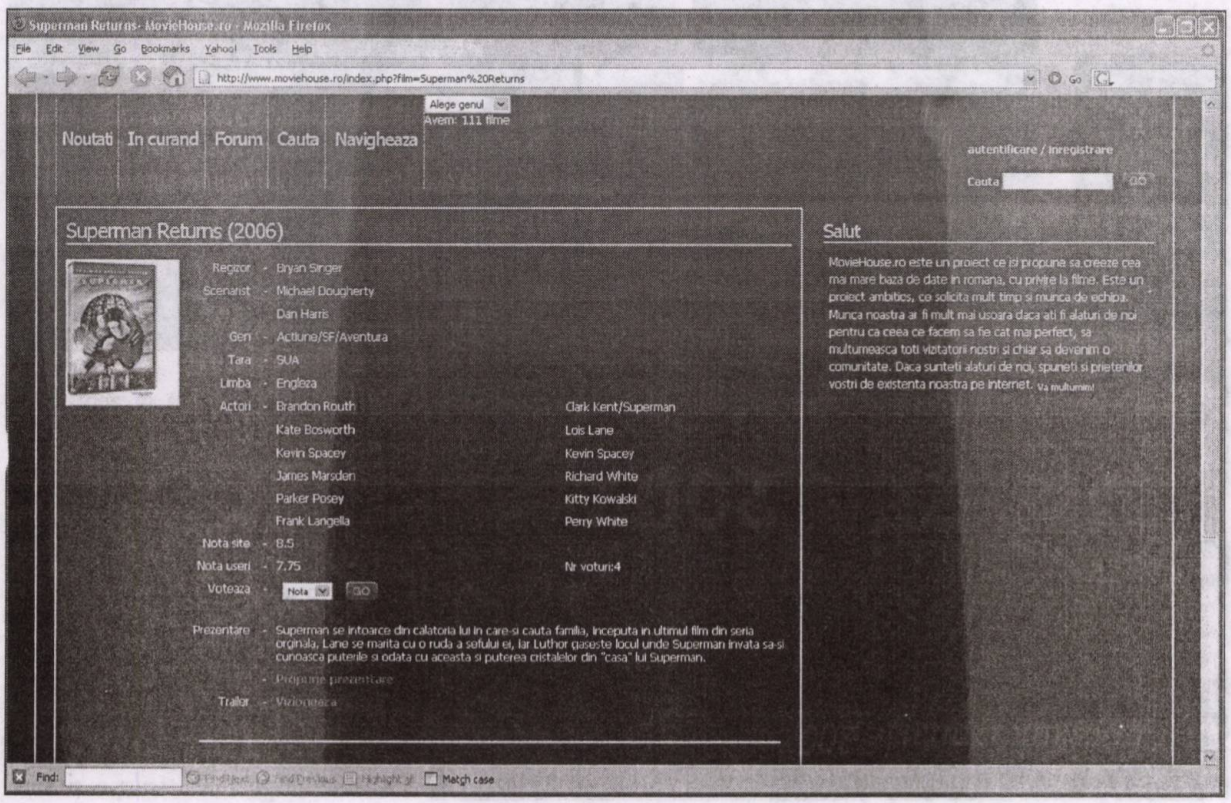
La ediția 2007 au fost înscrise peste 67 de lucrări, create de elevi din județele *Alba*, *Argeș*, *Bacău*, *Botoșani*, *Brăila*, *Cluj*, *Covasna*, *Galați*, *Gorj*, *Hunedoara*, *Ialomița*, *Mehedinți*, *Neamț*, *Olt*, *Sălaj*, *Suceava*, *Vâlcea*, *Vrancea* și din municipiul *București*.

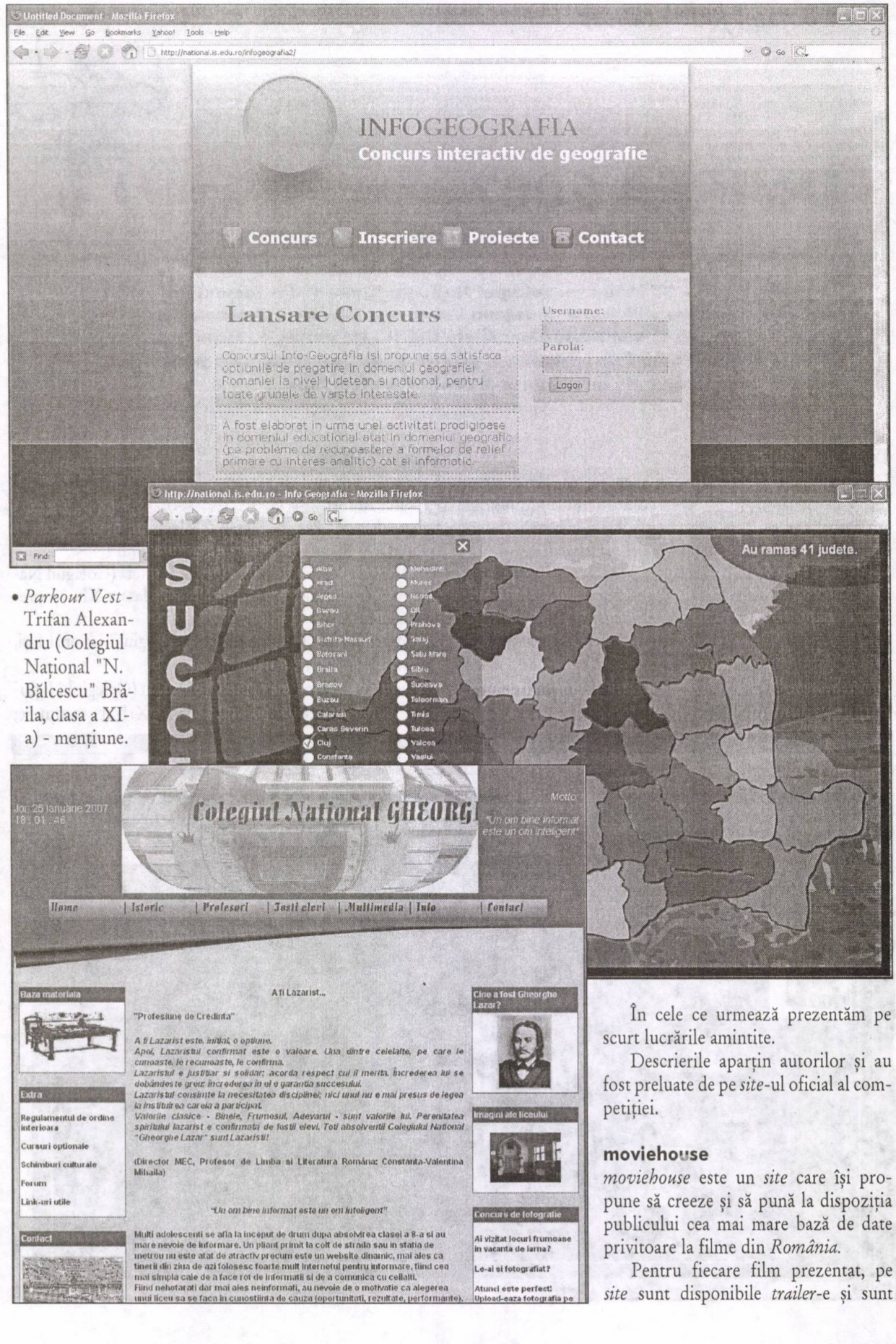
Pe lângă acest concurs a fost organizat unul de probleme de informatică și unul de matematică.

Pagini Web

Lucrările care au fost premiate la această secțiune a concursului au fost:

- *moviehouse* - Rotaru Ștefan (Colegiul Național "Ștefan cel Mare" Suceava, clasa a XI-a) - premiul I;
- *C.N. "Gheorghe Lazăr"* - Dulhaz Patricia (Colegiul Național "Gheorghe Lazăr" București, clasa a XI-a) - premiul al II-lea;
- *InfoGeografia* - Coșbuc Mircea (Colegiul Național Iași, clasa a XI-a) - premiul al III-lea;
- *Teenagers webchat* - Moscaliuc Ovidiu (Colegiul Național "Mihai Eminescu" Suceava, clasa a XII-a) - mențiune;





- *Parkour Vest* - Trifan Alexandru (Colegiul Național "N. Bălcescu" Brăila, clasa a XI-a) - mențiune.

În cele ce urmează prezentăm pe scurt lucrările amintite.

Descrierile aparțin autorilor și au fost preluate de pe *site*-ul oficial al competiției.

moviehouse

moviehouse este un *site* care își propune să creeze și să pună la dispoziția publicului cea mai mare bază de date privitoare la filme din *România*.

Pentru fiecare film prezentat, pe *site* sunt disponibile *trailer*-e și sunt

Teenagers webchat

Aceste este un *site Web*, care permite comunicarea în timp real între utilizatorii rețelei *Internet*.

Site-ul poate fi accesat *online* la adresa **http://195.238.80.9/chat2/**.

Parkour Vest

Aceste este un *site* sportiv informațional al echipei *parkour vest* din orașul *Brăila*, actualizat săptămânal sau lunar cu informații și material media nou.

prezentate comentarii, note discuții sau detalii despre actori, regizori, scenariști etc.

Site-ul poate fi accesat *online* la adresa **http://www.moviehouse.ro/**.

C.N. "Gheorghe Lazăr"

Acesta este un *site* de prezentare al Colegiului Național "Gheorghe Lazăr" din București.

InfoGeografia

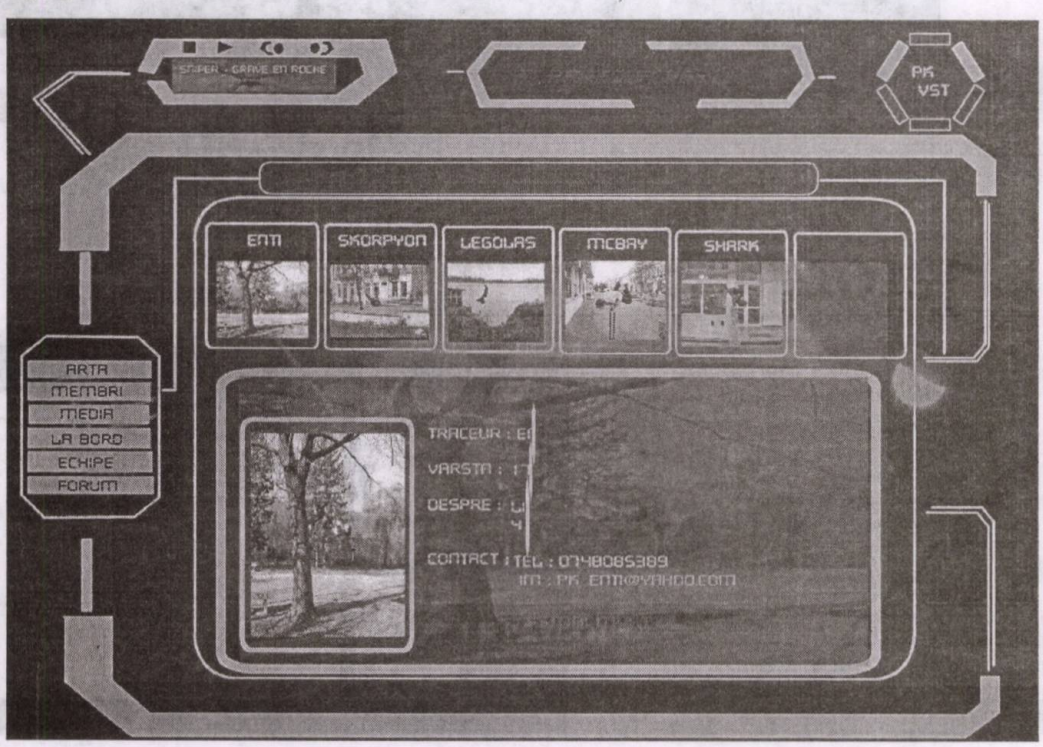
Acesta este *site*-ul pe care se va dezvolta viitorul concurs interactiv de geografie: *InfoGeografia*. *Site*-ul conține o parte dintre testele la care vor fi supuși elevii în perioada de concurs, la secțiunea: *Concurs/ antrenament*.

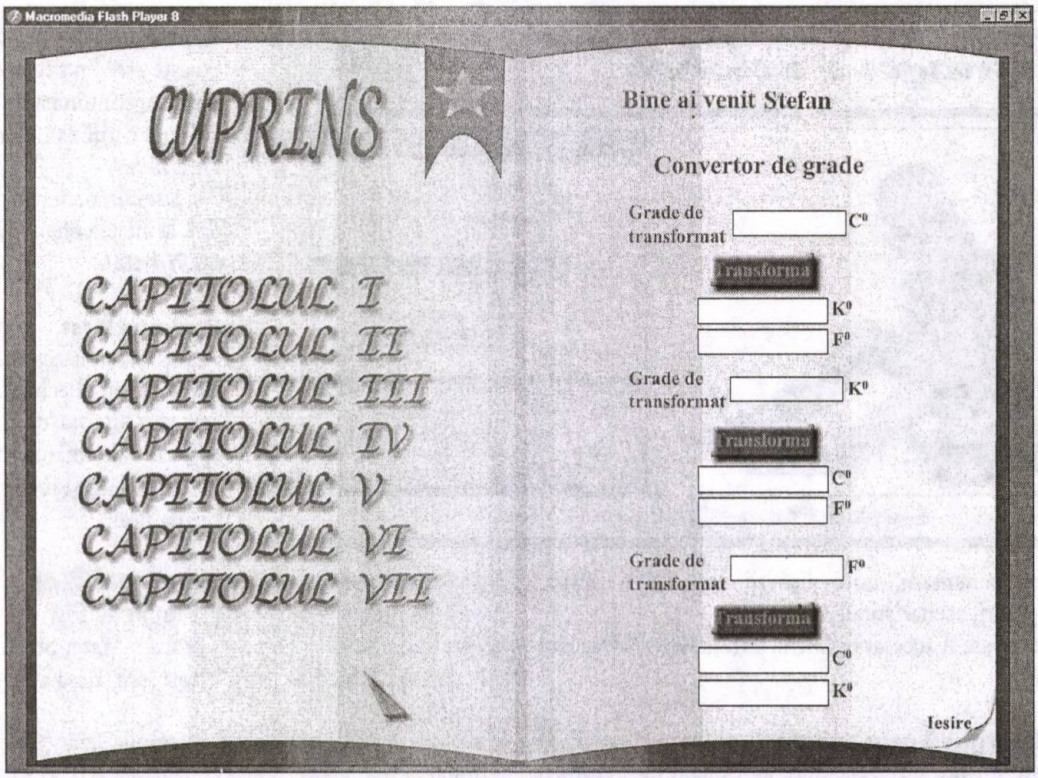
Site-ul poate fi accesat *online* la adresa **http://national.is.edu.ro/infogeografia2/**.

Software educațional și utilitar

Lucrările premiate la această secțiune au fost:

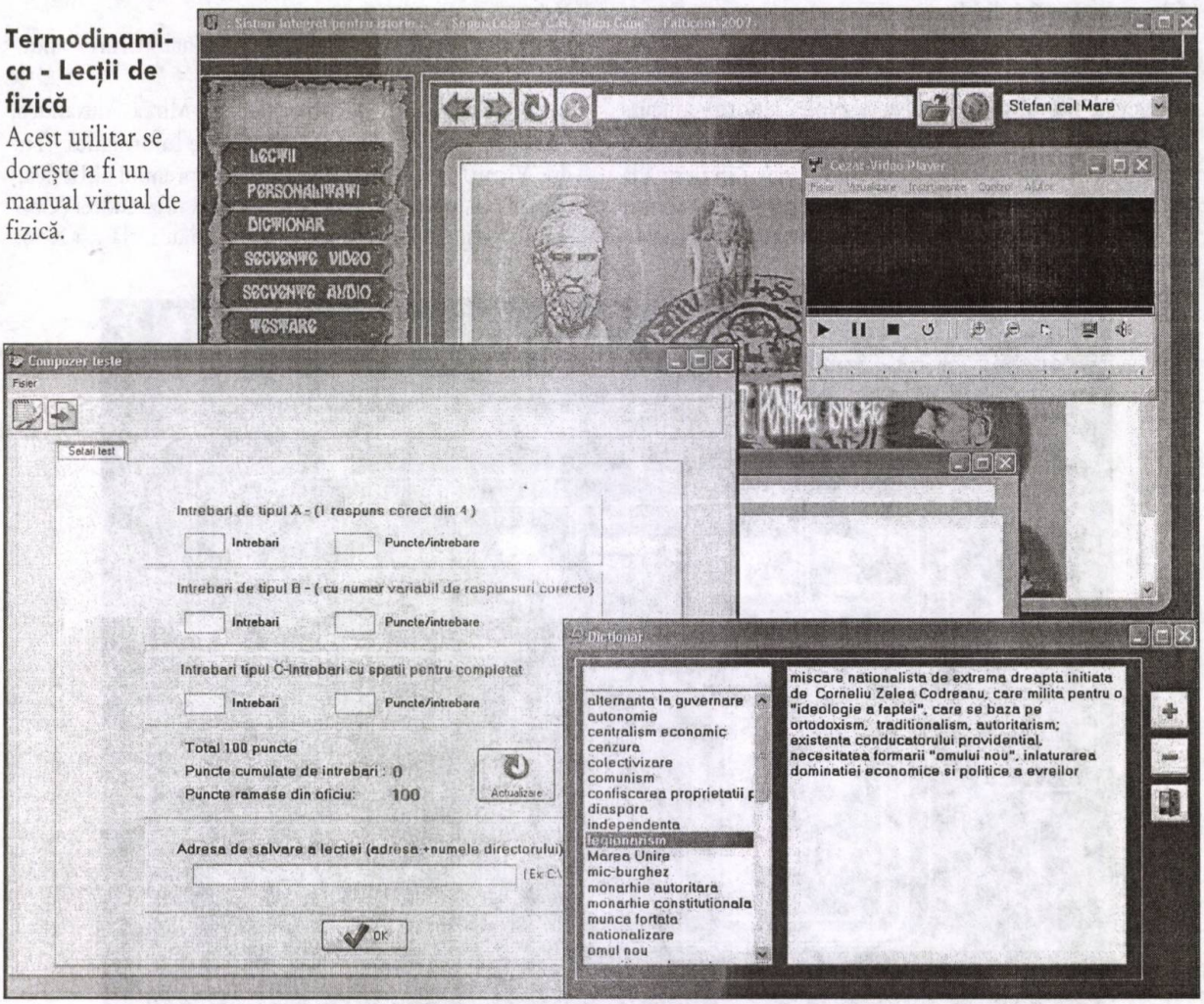
- *Termodinamica - Lecții de fizică* - Matei Ștefan (Liceul Pedagogic "Nicolae Iorga" Botoșani, clasa a XI-a) - premiul I;
- *Sistem integrat pentru istorie* - Sopon Cezar (Colegiul Național "Nicu Gane" Fălticeni, clasa a XII-a) - premiul al II-lea;
- *Grafix* - Chirilă Dan (Colegiul Național "Unirea" Focșani, clasa a XI-a) - premiul al III-lea;
- *Greedy euristic* - Copot Alexandru, Mirea Alexandru, Muscalu Mircea (Colegiul Național de Informatică "Tudor Vianu" București, clasa a XI-a) - premiul al III-lea;
- *Istoria lui Ștefan cel Mare* - Drîmbă Radu Andrei (Colegiul Național "E. Hurmuzachi" Rădăuți, clasa a X-a) - mențiune.

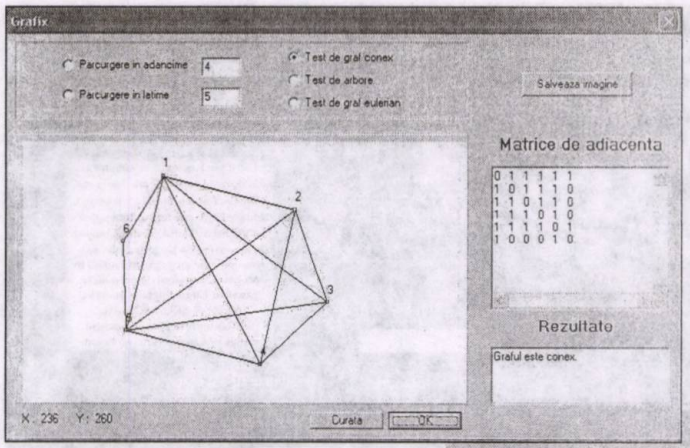




Termodinamica - Lecții de fizică

Acest utilitar se dorește a fi un manual virtual de fizică.





Grafix este deosebit de util pentru elevii care fac cunoștință cu grafurile, dar și pentru profesori. Aceștia din urmă pot evalua algoritmii implementați de elevi în clasă, generând un graf și rezultatul algoritmului propus, sau pot realiza subiecte pentru teste.

Greedy euristic

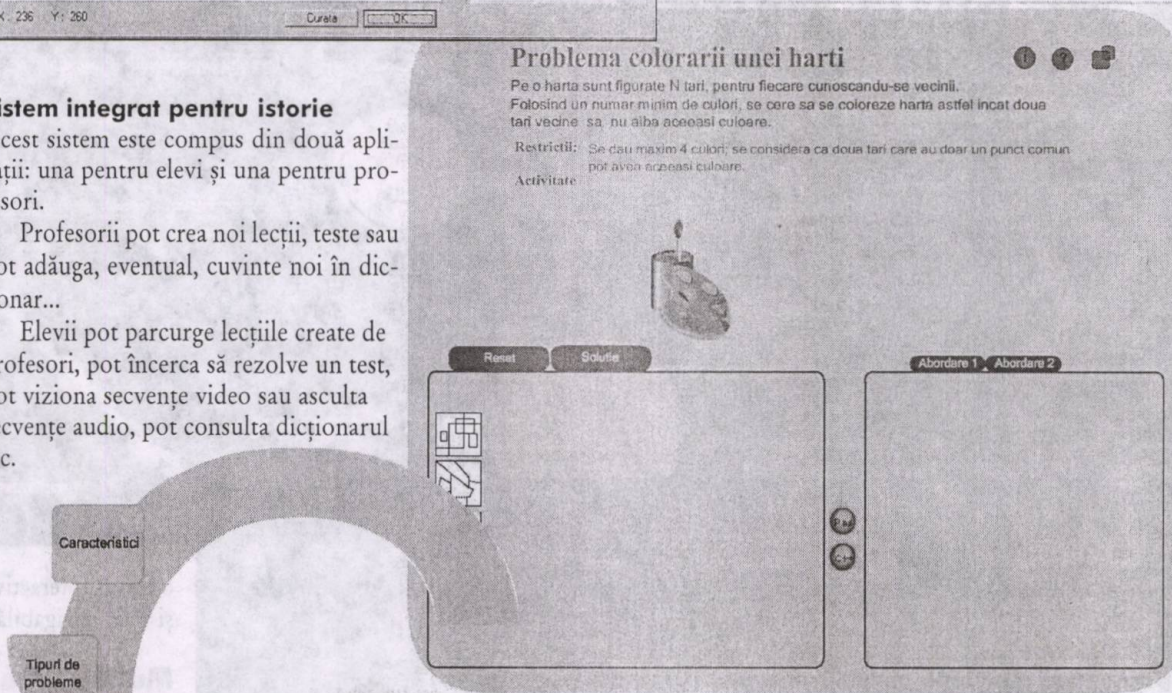
Proiectul prezintă algoritmul *Greedy* euristic într-un mod plăcut și interactiv, ceea ce duce la înțelegerea algoritmului cu ușurință de către pontențialii utilizatori.

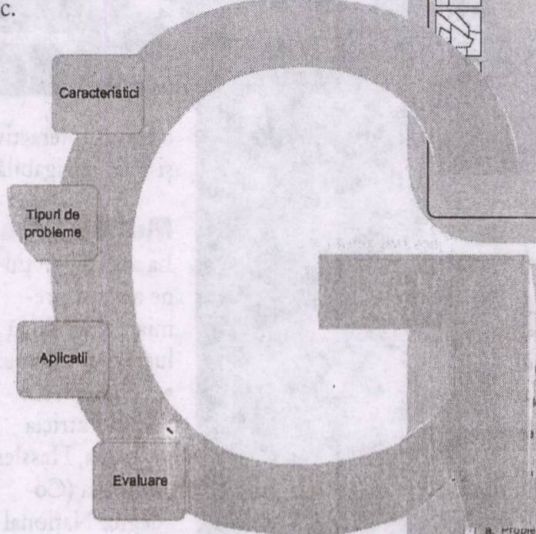
Sistem integrat pentru istorie

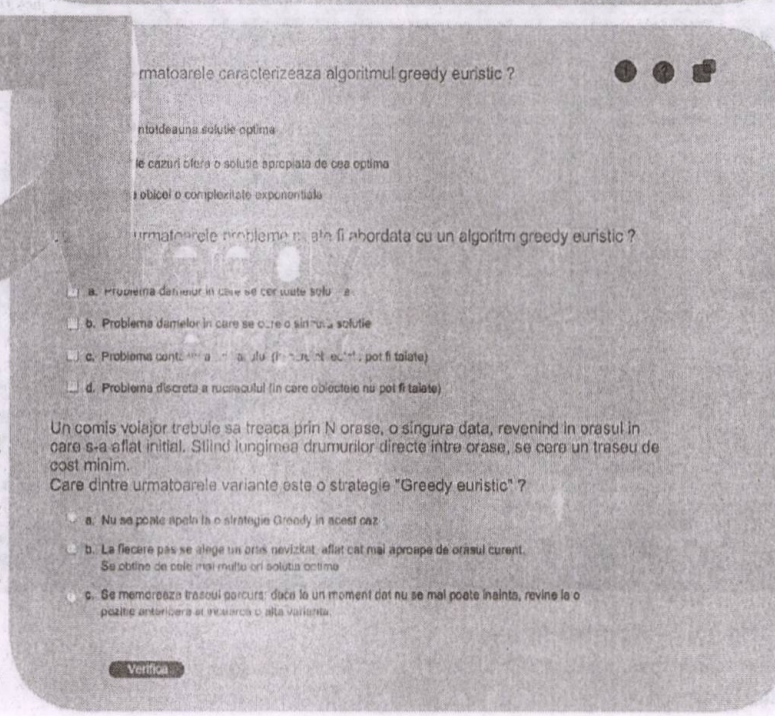
Acest sistem este compus din două aplicații: una pentru elevi și una pentru profesori.

Profesorii pot crea noi lecții, teste sau pot adăuga, eventual, cuvinte noi în dicționar...

Elevii pot parcurge lecțiile create de profesori, pot încerca să rezolve un test, pot viziona secvențe video sau asculta secvențe audio, pot consulta dicționarul etc.



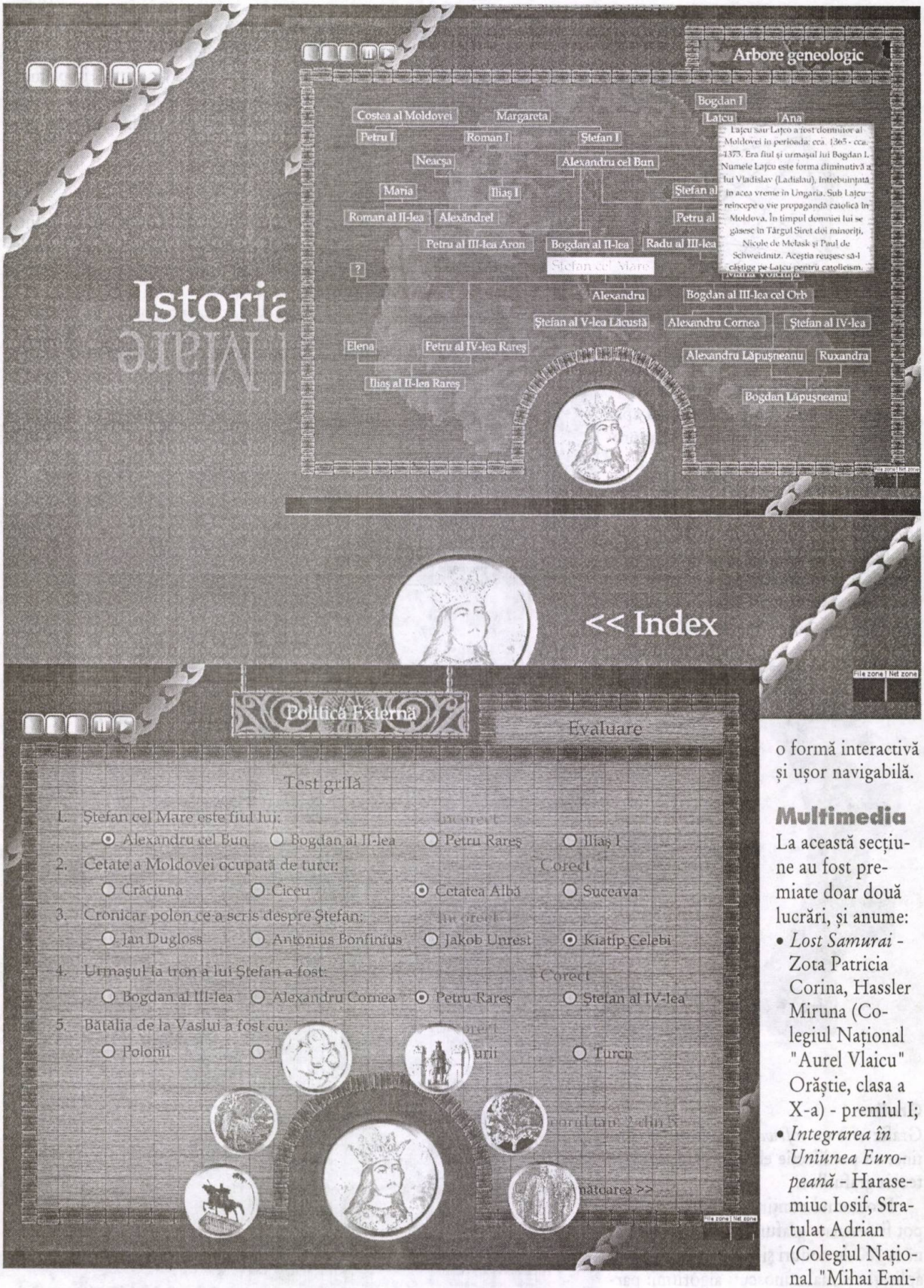




Grafix

Grafix este un *software* educațional destinat cu predilecție elevilor care studiază teoria grafurilor.

Programul conține o "tablă" pe care pot fi desenate grafuri neorientate cu până la 100 de noduri și asupra cărora pot fi aplicați cei mai cunoscuți algoritmi: parcurgere în lățime, parcurgere în adâncime, test de graf conex, test de arbore și test de graf eulerian.



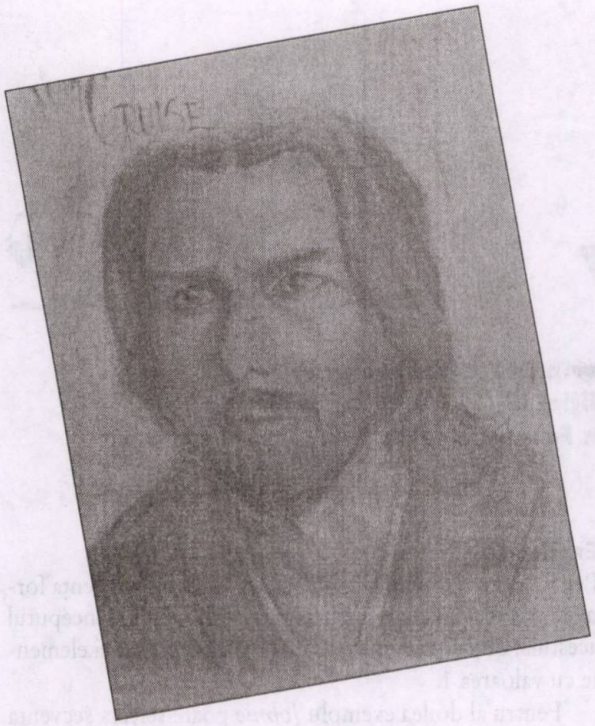
Istoria lui Ștefan cel Mare

Acest utilitar prezintă istoria lui *Ștefan cel Mare*, din mai multe perspective și puncte de vedere amplu detaliate, sub o formă interactivă și ușor navigabilă.

Multimedia

La această secțiune au fost premiate doar două lucrări, și anume:

- *Lost Samurai* - Zota Patricia Corina, Hassler Miruna (Colegiul Național "Aurel Vlaicu" Orăștie, clasa a X-a) - premiul I;
- *Integrarea în Uniunea Europeană* - Harasemiuc Iosif, Stratulat Adrian (Colegiul Național "Mihai Eminescu" Suceava, clasa a XI-a) - mențiune.



Ultimul samurai

Acest proiect prezintă o secvență din filmul *Ultimul samurai*, puțin personalizata...

Integrarea în Uniunea Europeană

Acest proiect prezintă pe larg momentul istoric al integrării *României* în *Uniunea Europeană*.

Celelalte secțiuni

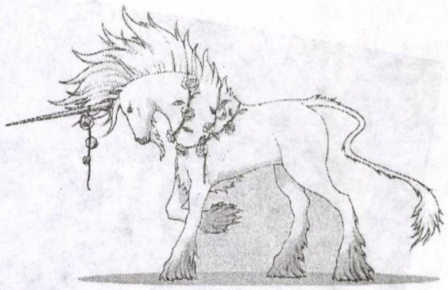
Problemele propuse la concursul de programare sunt prezentate în cadrul secțiunii *Probleme* a acestui număr.

Trebuie menționat faptul că acest concurs a fost organizat în colaborare cu echipa *InfoArena*, evaluarea fiind realizată folosind evaluatorul *online* pus la dispoziție de această echipă.

Concursul s-a desfășutar pe două categorii de vârstă (clasele a IX-a și a X-a, respctiv clasele a XI-a și a XII-a), la fiecare categorie fiind propuse câte două probleme.

Așa cum am spus, a existat și un concurs de matematică; la acesta au participat atât elevi de gimnaziu (clasele a VII-a și a VIII-a), cât și de liceu (clasele a IX-a, a X-a, a XI-a și a XII-a).

Focșani, ianuarie 2007



UNIREA 2007

Vă prezentăm în continuare enunțurile celor patru probleme propuse spre rezolvare la secțiunea de programare a ediției din acest an a concursului organizat de Colegiul Național "Unirea" din Focșani cu ocazia aniversării Unirii Principatelor.

Clasele a IX-a și a X-a

P070201: Maxsecv

Daniel Păsăilă

Johnie are un vector binar format din N elemente. El poate extrage o anumită subsecvență din vector, rămânând astfel cu un vector mai mic. Apoi, el poate insera subsecvența extrasă la orice poziție a vectorului rezultat.

Cerință

Să se determine lungimea maximă a unei subsecvențe pline de 1 pe care o poate obține *Johnie* dacă efectuează o singură operație.

Restricție

- $1 \le N \le 1000000$.

Date de intrare

Pe prima linie a fișierului **maxsecv.in** se află N, dimensiunea vectorului. Urmează apoi, pe următoarea linie, N valori 0 sau 1, reprezentând elementele vectorului.

Date de ieșire

Fișierul de ieșire **maxsecv.out** trebuie să conțină un singur număr, care reprezintă valoarea cerută.

Exemple

maxsecv.in
```
6
1 1 0 1 1 1
```

maxsecv.out
```
5
```

maxsecv.in
```
13
0 1 1 1 0 1 1 1 1 0 1 1 0
```

maxsecv.out
```
17
```

Explicație

Pentru primul exemplu *Johnie* poate muta subsecvența formată din ultimele trei elemente ale vectorului la începutul acestuia, obținând o subsecvență formată din cinci elemente cu valoarea 1.

Pentru al doilea exemplu *Johnie* poate selecta secvența de patru elemente cu valoarea 1 din interiorul vectorului și o poate muta lângă secvența de trei elemente cu valoarea 1 de la început.

Timp de execuție: 0.4 secunde/test

Memorie disponibilă: 16000 KB

P070202: Chernel

Adrian Vladu

Chernel a renunțat la afaceri și s-a apucat de matematică. De această dată, el studiază niște șiruri mai speciale.

Chernel pornește de la un șir format din N numere naturale $a_1, a_2, ..., a_N$, pe care aplică transformări succesive care schimbă un șir format din X elemente $a_1, a_2, ..., a_X$ în șirul format din X - 1 elemente $a_1 + a_2, a_2 + a_3, ..., a_{X-1} + a_X$.

Chernel repetă această operație până când rămâne cu un singur element.

De aici obține ***numărul caracteristic*** al șirului, ca fiind restul împărțirii acestui ultim element rămas la un număr natural M.

Chernel observă că valorile unor elemente din șirul inițial nu influențează numărul caracteristic al șirului; cu alte cuvinte, oricare ar fi valoarea acestora, numărul caracteristic rămâne același.

Cerință

Pentru N și M date, ajutați-l pe *Chernel* să determine câte valori care nu influențează numărul caracteristic există în șirul inițial.

Restricții

- $1 \le N \le 100000$;
- $1 \le M \le 1000000000$.

Date de intrare
Pe prima linie a fișierului de intrare **chernel.in** se află cele două numere naturale N și M.

Date de ieșite
Pe prima linie a fișierului de ieșire **chernel.out** se va afla un singur număr natural, reprezentând numărul elementelor din șirul inițial a căror valoare nu influențează numărul caracteristic al șirului.

Exemplu
chernel.in
```
3 2
```

chernel.out
```
1
```

Explicație
Fie șirul inițial a_1, a_2, a_3. Acesta se transformă în $a_1 + a_2, a_2 + a_3$, iar apoi în $a_1 + 2 \cdot a_2 + a_3$.

Pentru $M = 2$, singura valoare a șirului inițial care nu influențează numărul caracteristic este a_2.

Timp de execuție: 0.5 secunde/test

Memorie disponibilă: 16000 KB

Clasele a XI-a și a XII-a

P070203: Amenzi

Adrian Diaconu

Ion este polițist într-un mare oraș. În oraș există N intersecții legate între ele prin intermediul a M străzi pe care se poate circula în ambele sensuri. Pentru fiecare stradă se cunoaște timpul C_i necesar pentru a o parcurge.

Informat de o sursă sigură, *Ion* știe că în cursul următoarei zile se vor petrece K infracțiuni. Pentru fiecare infracțiune se cunosc următoarele date:
- T_i - momentul la care are loc infracțiunea;
- A_i - intersecția în care are loc infracțiunea;
- S_i - amenda pe care o poate da *Ion* dacă se află în intersecția A_i la momentul T_i.

Ion știe că în cursul zilei următoare trebuie să se întâlnească și cu soția sa, dar nu mai ține minte exact nici locul, nici ora la care trebuie să aibă loc întâlnirea.

Tot ce își amintește sunt P perechi de forma X_i, Y_i care indică faptul că s-ar putea să trebuiască să își întâlnească soția la momentul Y_i în intersecția X_i.

Cerință
Având la dispoziție toate aceste date, ajutați-l pe *Ion* să afle valoare maximă totală a amenzilor pe care le poate da pentru fiecare dintre cele P perechi pe care și le amintește.

Din momentul în care se întâlnește cu soția sa, *Ion* nu va mai da amenzi.

Restricții și precizări
- inițial *Ion* se află în intersecția 1, la momentul 0;
- timpii la care se petrec infracțiunile și la care *Ion* se poate întâlni cu soția sa fac parte din intervalul [0, 3500];
- $1 \le K \le 12000$;
- $1 \le P \le 8000$;
- $1 \le N \le 150$;
- $1 \le M \le 1500$;
- timpul necesar parcurgerii unei străzi este un număr întreg strict pozitiv;
- costul unei amenzi este un număr întreg care face parte din intervalul [1, 10000].

Date de intrare
Pe prima linie a fișierului de intrare **amenzi.in** se află patru numere întregi N, M, K și P, cu semnificația din enunțul problemei.

Următoarele M linii conțin câte trei numere a, b și c care indică faptul că există o stradă care leagă intersecțiile a și b și care poate fi parcursă în c unități de timp.

Următoarele K linii conțin câte trei numere a, t și s care indică faptul că va avea loc o infracțiune în intersecția a, la momentul t, care trebuie sancționată cu o amendă în valoare de c unități monetare.

Urmează P linii pe care se află câte două numere naturale x și y care indică faptul că este posibil ca *Ion* să trebuiască să își întâlnească soția în intersecția x la momentul y.

Date de ieșire
În fișierul de ieșire **amenzi.out** se vor afla P linii care vor conține sumele maxime pe care *Ion* le poate obține din amenzi în fiecare dintre cele P cazuri descrise în fișierul de intrare. Dacă *Ion* nu poate ajunge sub nicio formă în intersecția respectivă la momenul stabilit, scrieți valoarea -1 pe testul respectiv.

Exemplu
amenzi.in
```
5 7 4 2
5 4 4
4 3 5
2 3 7
3 1 3
5 2 3
4 1 10
4 2 1
2 6 5736
2 20 2567
5 6 1530
3 3 4067
1 50
3 15
```

amenzi.out
```
6634
4067
```

Explicație

Pentru primul caz *Ion* se va deplasa în intersecția 3, unde va ajunge la momentul 3, exact la timp pentru a da o amendă în valoare de 4067. Apoi se deplasează în intersecția 2, unde ajunge la momentul 10. Aici așteaptă până la momentul 20 când dă o amendă în valoare de 2567. El se deplasează apoi la intersecția 1, unde își va astepta soția până la momentul 50.

Pentru al doilea caz *Ion* se deplasează în intersecția 3, unde ajunge la momentul 3, dă amenda în valoare de 4067, apoi își așteaptă soția până la momentul 15.

Timp de execuție: 0.3 secunde/test

Memorie disponibilă: 16000 KB

P070204: Secvența 5

Mircea Pașoi

Zăhărel are de rezolvat încă o problemă cu secvențe! De data aceasta el are un șir format din N numere naturale și dorește să știe câte subsecvențe care conțin între L și U elemente distincte există în șirul respectiv.

Restricții și precizări

- $1 \le L \le U \le N \le 2^{20}$;
- valoarea fiecărui element din șir este un număr întreg cuprins în intervalul $[1, 2^{32} - 1]$;
- o subsecvență este un subșir de elemente care apar pe poziții consecutive în șirul inițial.

Date de intrare

Prima linie a fișierului de intrare **secv5.in** conține numerele naturale N, L și U.

Următoarele N linii vor conține câte un număr natural fiecare reprezentând un element al șirului.

Date de ieșire

Pe prima linie a fișierului **secv5.out** se va afișa numărul subsecvențelor care conțin între L și U elemente distincte.

Exemplu

secv5.out

```
5 2 3
13
13
7
9
9
```

secv5.out

```
8
```

Explicație

Cele opt subsecvențe sunt:

- 13 13 7
- 13 13 7 9
- 13 13 7 9 9
- 13 7
- 13 7 9
- 13 7 9 9
- 7 9
- 7 9 9

Timp de execuție: 1.5 secunde/test

Memorie disponibilă: 65536 KB

Clasele a IX-a și a X-a

1. Gheorghe Cosmin	200
2. Marius Drăguș	190
2. Cezar Mocan	190
2. Adriana Sperlea	190
2. Bogdan Tătăroiu	190
2. Marian Gogu	190
7. Bogdan Casu-Pop	160
8. Vlad Duta	140
9. Sergiu Vicol	130
9. Dan-George Filimon	130
9. Cristina-Elena Petrovici	130
9. Andrei Stănică	130

Rezultatele sunt preluate de la: http://infoarena.ro/unirea-2007/rezultate-9-10

Clasele a XI-a și a XII-a

1. Valentin Stanciu	180
2. Ștefan Filip	170
3. Andrei Blănaru	160
3. Cristina Stancu-Mara	160
5. Teodor Roșu	150
6. Adrian Airinei	110
7. Andrei Grigorean	100
7. Constantin Berzan	100
7. Vlad Dumitriu	100
7. Marian Gogu	100

Rezultatele sunt preluate de la: http://infoarena.ro/unirea-2007/rezultate-11-12

Șansa de a deveni CAMPION

.Campion, competiția care formează viitorii campioni ai României

Orice societate informatizată are nevoie de tot mai mulți specialiști în domeniul informaticii și al tehnologiei informației. Tinerii din România și Europa de Sud-Est s-au dovedit competitivi în aceste domenii, atât în țară, cât și în străinătate, prin rezultatele obținute la competiții internaționale, prin proiectele de cercetare susținute în universități, prin produsele dezvoltate la firmele la care lucrează.

România și Europa de Sud Est poate deveni una dintre principalele țări producătoare și exportatoare de software și în acest sens, pregătirea de performanță a elevilor în domeniul informaticii devine o necesitate. Această pregătire trebuie dublată de stimularea spiritului competitiv, dar și de formarea abilităților de a iniția și dezvolta o afacere proprie în domeniul IT.

.Campion este un concurs anual, organizat sub egida Centrului de Excelență SIVECO în colaborare cu Ministerul Educației și Cercetării, care are ca obiectiv principal pregătirea de performanță a elevilor în informatică. Acest obiectiv urmărim să îl atingem prin dezvoltarea unei gândiri algoritmice, a abilităților de programare, a spiritului competitiv și a abilităților de inserare socio-profesională a participanților.

Elevii sunt împărțiți în 3 grupe de pregătire, în concordanță cu nivelul/vârsta lor.

Grupa S (Small) - începători în informatică, primul an în domeniul programării; vârsta nu trebuie să depășească 16 ani (clasa a IX-a sau elevi de gimnaziu);

Grupa M (Medium) - nivel intermediar, un an sau doi ani experiență în domeniul programării; vârsta nu trebuie să depășească 17 ani (clasa a X-a)

Grupa L (Large) - avansați în programare (clasele XI-XII).

Un elev poate participa la pregătire la grupa la care se încadrează sau la orice altă grupă superioară. Pregătirea se desfășoară în runde, runde de pregătire și runde de concurs, alternativ. Într-o rundă un elev primește spre rezolvare 2 probleme. O rundă de pregătire durează de regulă 10 zile. În rundele de concurs concurenții participă la un concurs online. Rundele de concurs se desfășoară sâmbăta și reprezintă o olimpiadă desfășurată online.

Toate soluțiile trimise sunt evaluate cu soft de evaluare automată, în condiții similare cu cele existente la olimpiadele internaționale de informatică.

Odată cu rezultatele evaluării, autorii problemelor postează pe site o soluție posibilă pentru problemele propuse, precum și grilele utilizate pentru evaluare.

Concursul .Campion este coordonat de profesorii Emanuela Cerchez și Marinel Șerban de la Liceul de Informatica "Grigore Moisil"din Iași.

La începutul ediției 2006-2007 s-au înscris în concurs la grupa mică (clasa a IX-a sau elevi de gimnaziu) 1081 elevi, la grupa mijlocie (clasa a X-a) 870 elevi, iar la grupa mare (clasele XI-XII) 839 elevi.

"Prin competiția .Campion, desfășurată sub umbrela Centrului Virtual de Excelență SIVECO Romania ne dorim să oferim tuturor elevilor șanse egale să participe la o pregătire de performanță în domeniul științei informaticii , de a le stimula pasiunea pentru acest domeniu și se pare ca am reușit. În perioada 2003-2006, tineri din întreaga țară au muncit cu dăruire și și-au dovedit valoarea în concursuri. O dovedesc rezultatele obținute de finaliștii .Campion la concursurile naționale și internaționale" declară Radu Jugureanu, eContent Manager în cadrul SIVECO Romania

Ediția a V-a a concursului a început la 1 nov 2006 și constă din 12 runde, ultima rundă fiind și suportul concursului național de informatică Urmașii lui Moisil organizat de Liceul de informatică Grigore Moisil din Iași în perioada 23-25 martie 2007.

Site-ul oficial al concursului este: http://campion.edu.ro/

Dorim succes în continuare elevilor la olimpiadele și concursurile cu care se confruntă anul acesta.

Emanuela Cerchez

POKER

Mihai Scorţaru

În cadrul acestui articol vom prezenta modul în care poate fi creat un applet Java care estimează şansele de câştig ale participanţilor la o mână a unui joc de hold'em poker.

Regulile jocului

O mână de *hold'em poker* se desfăşoară, în principiu, astfel:

- se pariază *blind*-urile;
- fiecare jucător primeşte două cărţi (fiecare jucător vede doar propriile cărţi);
- prima rundă de pariuri;
- pe masă sunt arătate trei cărţi (*flop*-ul);
- a doua rundă de pariuri;
- pe masă este arătată o carte (*turn*-ul);
- a treia rundă de pariuri;
- pe masă este arătată o carte (*river*-ul);
- ultima rundă de pariuri;
- acordarea premiului.

Jocul se poate întrerupe la oricare dintre rundele de pariuri dacă cineva pariază şi nimeni nu "intră".

La sfârşitul jocului, fiecare jucător are la dispoziţie şapte cărţi (două proprii şi cinci pe masă). Dintre acestea el alege cinci astfel încât să aibă o mână cât mai bună.

Mâinile sunt ordonate astfel:

- chintă royală (*royal flush*);
- chintă de culoare (*straight flush*);
- careu (*four of a kind*);
- ful (*full house*);
- culoare (*flush*);
- chintă (*straight*);
- trei bucăţi (*three of a kind*);
- două perechi (*two pairs*);
- o pereche (*one pair*);
- carte mare (*high card*).

Este destul de clar că o mână dintr-o anumită categorie va "bate" orice dintr-o categorie inferioară.

În cele ce urmează vom prezenta fiecare categorie şi vom descrie modul în care are loc departajarea în cadrul aceleiaşi categorii.

Royal flush

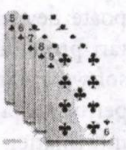
O chintă royală este formată dintr-un decar, un valet, o damă, o rigă (un popă) şi un as, toate având aceeaşi culoare.

La jocul *hold'em poker* nu există posibilitatea ca doi jucători să aibă chinte royale diferite.

Există posibilitatea ca această chintă să se afle pe masă, caz în care toţi jucătorii rămaşi câştigă (potul se împarte).

Straight flush

O chintă de culoare este formată din cinci cărţi cu valori consecutive care au aceeaşi culoare.



În cazul în care mai mulţi jucători au chinte de culoare, departajarea se realizează în funcţie de cartea cu cea mai mare valoare.

De exemplu, dacă pe masă se află un optar, un nouar, un decar şi un valet de cupă, primul jucător are un şeptar de cupă, iar al doilea are o damă de cupă, atunci primul jucător are o chintă de culoare la valet în timp ce al doilea are o chintă de culoare la damă; al doilea jucător va câştiga.

Trebuie menţionat faptul că, practic, o chintă royală este o chintă de culoare la as; din acest motiv, în implementare, nu vom face distincţie între chinta de culoare şi chinta royală; evident, chinta de culoare la as va "bate" orice alt tip de chintă de culoare.

Four of a kind

Un careu este format din patru cărţi care au aceeaşi valoare. Cea de-a cincea carte poate avea orice altă valoare.

Există posibilitatea ca doi jucători să aibă careuri diferite. De exemplu, fiecare ar putea avea în mână o pereche, iar pe masă, printre cele cinci cărţi, s-ar putea afla celelalte două cărţi cu aceeaşi valoare. Careurile diferite sunt departajate în funcţie de valoarea cărţilor care le formează.

De asemenea, există posibilitatea ca doi jucători să aibă careuri de aceeași valoare (careul este pe masă). În această situație departajarea se va face în funcție de valoarea celei de-a cincea cărți.

Full house

Un ful este format din cinci cărți dintre care trei au aceeași valoare, iar celelalte două au o altă valoare.

În cazul în care mai mulți jucător au ful, departajarea se realizează în funcție de cele trei cărți cu valori identice.

Dacă egalitatea persistă, departajarea se realizează în funcție de valorile celorlalte două cărți.

Flush

O culoare este formată din cinci cărți care au aceeași culoare, dar nu sunt consecutive (ar deveni o chintă de culoare).

La jocul *hold'em poker* nu există posibilitatea ca doi jucători să aibă culori din suite diferite (de exemplu trefle și pici).

Totuși, jucătorii pot avea culori diferite din aceeași suită. În acest caz departajarea se realizează în funcție de cartea cu valoarea cea mai mare. Dacă egalitatea persistă se trece la cartea cu valoarea imediat inferioară. Procedeul continuă până în momentul în care ajungem la cărți cu valori diferite sau la ultima carte.

Să presupunem cu un jucător are un șeptar și un valet de roșu, celălalt are un optar și un decar de roșu, iar pe masă avem un doiar, o damă și un as de roșu. Ambii jucători au culoare, dar primul câștigă deoarece a treia sa carte ca valoare este mai mare (valet față de decar).

Straight

O chintă este formată din cinci cărți cu valori consecutive care nu au toate aceeași culoare.

În cazul în care mai mulți jucători au chinte, departajarea se realizează în funcție de cartea cu cea mai mare valoare.

De exemplu, dacă pe masă se afla un șeptar, un nouar, un decar și un valet, primul jucător are un optar, iar al doilea are o damă, atunci primul jucător are o chintă la valet, în timp ce al doilea are o chintă la damă; al doilea jucător va câștiga.

Three of a kind

Avem trei bucăți dacă trei cărți au aceeași valoare, iar celelalte două cărți au valori diferite între ele și diferite de valorile celorlalte trei (în caz contrar am avea ful sau careu).

Există posibilitatea ca doi jucători să aibă trei bucăți diferite. În această situație departajarea se face în funcție de valoarea cărților.

De asemenea, este posibil ca doi sau mai mulți jucători să aibă trei bucăți cu aceeași valoare. În acest caz departajarea se face în funcție de cea mai mare dintre celalate două cărți. Dacă egalitatea persistă se realizează departajarea în funcție de cea mai mică dintre celelalte două cărți.

Two pairs

Avem două perechi dacă două dintre cărți au aceeași valoare, alte două cărți au o altă valoare diferită de cea a primelor două, iar cea de-a cincea carte are o valoare diferită de valorile primelor patru.

Dacă mai mulți jucători au două perechi, departajarea se face în funcție de valoarea perechii cu valoarea mai mare. Dacă egalitatea persistă, departajarea se face în funcție de valoarea perechii cu valoarea mai mică. În cazul în care avem în continuare egalitate, departajarea are loc în funcție de valoarea celei de-a cincea cărți.

One pair

Avem o pereche dacă două cărți au aceeași valoare, iar celelalte trei au valori diferite între ele și diferite de valorile primelor două cărți.

Dacă mai mulți jucători au o pereche departajarea se face în funcție de valoarea cărților care formează perechea. Dacă egalitatea persistă putem departaja jucătorii în funcție de valoarea celei mai mari dintre cele trei cărți rămase. Egalitatea poate persista, departajarea realizându-se în funcție de cea de-a doua și apoi de cea de-a treia carte ca valoare.

High card

În cazul în care o mână nu intră în niciuna dintre categoriile anterioare, departajarea se realizează în funcție de cartea cu valoarea cea mai mare. Dacă egalitatea persistă se trece la cartea cu valoarea imediat inferioară. Procedeul continuă până în momentul în care ajungem la cărți cu valori diferite sau la ultima carte.

Prezentarea aplicației

Vom crea un *applet* care va permite utilizatorilor să estimeze șansele pe care le au doi jucători de a câștiga o mână în funcție de "situația curentă".

Utilizatorul va putea selecta cărțile celor doi jucători și, eventual, cărțile de la flop sau cartea de la turn. Nu va fi permisă și alegerea riverului, deoarece după alegerea acestuia nu mai trebuie estimate șanse, fiind evident cine a câștigat mâna.

După alegerea cărților, se poate cere estimarea șanselor pe baza acestor alegeri. Șansele vor fi exprimate în procente; vor fi prezentate șansele ca victoria să aparțină primului jucător, șansele ca victoria să aparțină celui de-al doilea jucător, precum și șansele de a avea egalitate.

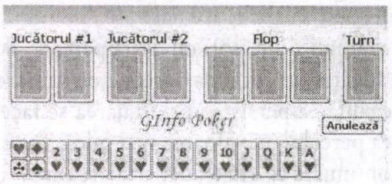
Butonul care permite calculul șanselor va fi activ doar dacă sunt selectate cele patru cărți ale jucătorilor și nu avem nicio carte a flopului sau avem toate cărțile flopului (putem sau nu avea și turnul).

În momentul alegerii unei cărți, aceasta va dispărea pentru a nu mai putea fi aleasă încă o dată.

Ultima carte aleasă poate fi eliminată prin efectuarea unui *click* pe cartea respectivă.

Va exista, de asemenea, un buton pentru anularea alegerii curente (resetare).

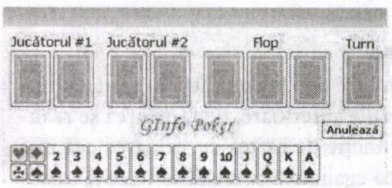
În imaginea următoare este prezentată configurația inițială a ferestrei *applet*-ului.



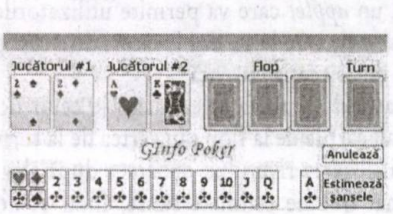
Această imagine este păstrată într-un fișier care va fi încărcat (**`background.jpg`**).

În colțul din stânga-jos avem simbolurile celor patru suite (cupe, pici, trefle și carouri). Prin efectuarea unui *click* pe unul dintre cele patru simboluri, cărțile care pot fi alese se schimbă.

Avem alte trei imagini care ne permit afișarea noilor variante de alegere (**`diamonds.jpg`**, **`spades.jpg`** și **`clubs.jpg`**). În momentul schimbării suitei, cele 13 cărți din partea de jos a ferestrei vor fi înlocuite. În continuare prezentăm fereastra *applet*-ului după ce au fost alese picile.



Putem să alegem acum cărți pentru cei doi jucători. Cărțile selectate vor dispărea. După alegerea cărților pentru primii doi jucători, va apărea butonul care permite estimarea șanselor:



Putem cere acum o estimare a șanselor. Rezultatul va fi afișat în zona liberă din stânga:

Observăm acum un lucru destul de interesant: deși un as și o rigă par a fi cărți puternice, ele au șanse mai mici decât o pereche de a câștiga mâna (chiar dacă avem cea mai mică pereche posibilă).

Dacă renunțăm la ultima carte aleasă (efectuăm un *click* pe ea), butonul care permite estimarea șanselor va dispărea:

Observăm că după renunțarea la riga de treflă, aceasta poate fi din nou selectată.

Revenim la alegerea făcută (reselectăm riga de treflă - nu avem buton de *Undo*) și trecem la alegerea flopului. Până în momentul în care nu au fost alese toate cele trei cărți, nu este vizibil butonul care permite estimarea șanselor:

După alegerea ultimei cărți a flopului, butonul devine vizibil și pot fi estimate șansele:

Cei doi jucători au fost destul de norocoși... Primul are deja trei bucăți, iar al doilea are două perechi. Totuși, primul are deja șanse mult mai mari. Al doilea jucători va câștiga doar dacă la turn sau la river vom avea un as sau o rigă și nu am avea un doiar. Astfel el ar obține un ful și, deși primul jucător ar avea și el ful, mâna celui de-al doilea va fi mai puternică.

Putem alege acum turnul și cere o nouă și ultimă evaluare a șanselor. Dacă avem un doiar la turn, primul jucător a obținut un careu, iar al doilea nu mai are nicio șansă de a câștiga.

Dacă avem la turn un as sau o rigă atunci, deși cel deal doilea jucător are șanse foarte mari, primul mai poate spera la un doiar norocos la river; șansele ca acesta să apară sunt de una din 44 (în pachet au rămas 44 de cărți, în afară de cele opt alese), adică aproximativ 2.27%.

Pregătirea implementării

Așadar, vom încerca să creăm un *applet* care să se comporte așa cum am prezentat.

În primul rând vom avea nevoie de codul *HTML* care permite afișarea *applet*-ului. Acesta este foarte simplu:

```
<HTML>
  <HEAD>
    <TITLE>
      Poker
    </TITLE>
  </HEAD>
    <APPLET
      code = "Poker"
      name = "Poker"
      width = 385
      height = 180
      align = top>
    </APPLET>
  </BODY>
</HTML>
```

Mai avem nevoie de imaginile menționate (**background.jpg**, **diamonds.jpg**, **spades.jpg** și **clubs.jpg**), precum și de imaginile cărților.

Vom avea 52 de imagini pentru cărți, toate fiind fișiere **.jpg**. Fiecare fișier va avea un nume format din valoarea cărții și culoarea acesteia. Valorile vor fi indicate prin 2, 3, 4, 5, 6, 7, 8, 9, 10, j, q, k sau a, iar culorile prin c (*clubs - trefle*), d (*diamonds - carouri*), h (*hearts - cupe*) sau s (*spades - pici*). De exemplu, decarul de cupă va fi păstrat în fișierul **10h.jpg**, șeptarul de treflă în fișierul **7c.jpg**, dama de caro în fișierul **qd.jpg**, iar asul de pică în fișierul **as.jpg**.

Nu vor fi încărcate toate imaginile, ci doar pe măsură ce este nevoie de ele. Imaginea **background.jpg** conține, practic, întreaga interfață:

Cărțile care nu mai pot fi selectate vor fi șterse prin desenarea unui dreptunghi alb. De asemenea, butonul care permite estimarea șanselor va fi șters, dacă este cazul, tot prin desenarea unui dreptunghi alb.

Practic, peste această imagine vor fi plasate alte imagini (cele ale unor cărți sau cele care permit afișarea în partea de jos a carourilor, picilor sau treflelor), dacă este cazul, vor fi desenate dreptunghiuri albe sau vor fi scrise rezultatele estimărilor.

Metoda `paint()` a *applet*-ului va desena mai întâi imaginea de fundal, urmând ca apoi, în funcție de starea curentă să deseneze sau să șteargă alte elemente.

Pentru estimarea șanselor nu vom utiliza o metodă exhaustivă (generarea tuturor combinațiilor posibile), ci una probabilistică. Vom genera un număr de configurații și vom calcula astfel o estimare. Implementarea poate fi modificată relativ ușor pentru a genera toate configurațiile posibile și a obține un rezultat exact.

Datorită acestui mod de calcul, putem avea mici erori ale rezultatelor. De exemplu, este evident că în situația de mai jos cei doi jucători au șanse egale. Totuși, *applet*-ul a afișat o diferență de 0.1%.

Nu vom avea cu siguranță egalitate în situația din imaginea de mai sus deoarece este posibil (aproape 15% șanse), ca unul dintre cei doi jucători să obțină o culoare (sau chiar o chintă royală).

Dacă numărul configurațiilor studiate este suficient de mare, aceste erori vor fi foarte mici. Oricum, pe puțini îi interesează șansele cu o precizie foarte mare. În general, jucătorii ar dori să știe "cam câte șanse au într-o anumită situație".

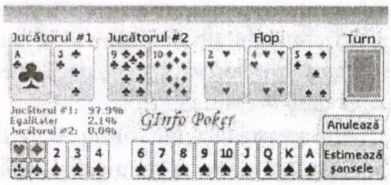
Pentru a genera o configurație, vom alege pentru fiecare dintre cărțile necunoscute, aleator o carte nefolosită încă. Vom avea astfel un număr total de nouă cărți: cinci pe masă și câte două pentru fiecare jucător.

Fiecare jucător are la dispoziție șapte cărți pentru a-și alege mâna. Vom încerca, pe rând să stabilim dacă mâna face parte dintr-una dintre categoriile amintite la începutul articolului, în ordinea corespunzătoare

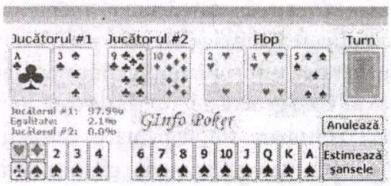
Dacă numai un jucător are mâna într-o anumită categorie, ne vom opri și îl vom "declara" câștigător. Dacă ambii jucători au mâna în aceeași categorie, atunci vom încerca să îi departajăm pe baza criteriilor menționate.

Pentru a simplifica operațiile care trebuie efectuate, vom ordona crescător cărțile fiecărui jucător.

Mai trebuie menționat faptul că va trebui să avem grijă la chintă (este posibilă o chintă formată din as, doiar, treiar, pătrar și cincear, asul având valoarea 1 în acest caz).

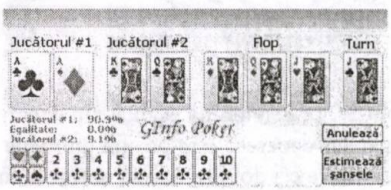


Asul este singura carte care poate avea două valori diferite, în funcție de situație. Nu este posibil, de exemplu, să avem o chintă de forma damă, rigă, as, doiar, treiar.



În situația de mai sus se poate ajunge la egalitate dacă vom obține o chintă pe masă (un treiar, urmat apoi de un as sau de un șesar; evident, și în ordine inversă - treiarul la river).

Trebuie menționat faptul că putem folosi doar cinci din cele șapte cărți. Cu șapte cărți am putea avea trei perechi, de două ori câte trei bucăți (în acest caz avem, în realitate, un ful), sau chiar două fuluri.



În situația de mai sus primul jucător are două perechi (ași și valeți) și o rigă, în timp ce al doilea jucător are tot două perechi (rigi și dame) și un valet; șanse mai mari are primul jucător; al doilea va câștiga doar dacă la river vom avea o damă sau o rigă, ceea ce ar duce la un ful. Dacă la river am avea un alt valet, atunci primul jucător ar avea un ful de valeți cuași, iar al doilea ar avea un ful de valeți cu rigi (ar câștiga primul jucător).

Mai trebuie menționat faptul că fiecare carte va fi codificată printr-un număr cuprins între 0 și 51, obținut prin înmulțirea cu 4 a valorii cărții (număr cuprins între 0 și 12) și adunarea la rezultat a culorii (număr cuprins între 0 și 3).

Implementarea

Prezentăm în continuare scheletul implementării *applet*-ului (corpurile metodelor vor fi prezentat ulterior):

```
import java.applet.Applet;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;
import java.awt.*;
import java.net.URL;
import java.net.MalformedURLException;
import java.util.Random;

public class Poker extends Applet implements
                ActionListener, MouseListener
{
    public static final int MAX = 1000000;

    public static final int HEARTS = 0;
    public static final int SPADES = 1;
    public static final int CLUBS = 2;
    public static final int DIAMONDS = 3;

    public static final int
                        CHANGE_TO_HEARTS = 10;
    public static final int
                         CHANGE_TO_CLUBS = 11;
    public static final int
                        CHANGE_TO_SPADES = 12;
    public static final int
                     CHANGE_TO_DIAMONDS = 13;
    public static final int
                       CARD_SELECTED = 14;
    public static final int
                        CARD_REMOVED = 15;
    public static final int RESET = 16;
    public static final int COMPUTE = 17;

    public static final int FIRST = 21;
    public static final int SECOND = 22;
    public static final int TIE = 23;

    private int mainSuit = HEARTS;

    private boolean used[];
    private int selectedCards[];
    private int noSelected;
    private String values[] =
        {"2", "3", "4", "5", "6", "7", "8",
          "9", "10", "J", "Q", "K", "A"};
    private String colors[] =
                        {"h", "s", "c", "d"};
    private Image cards[];

    private Image background;
    private Image spades;
    private Image diamonds;
    private Image clubs;

    private Font symbol;
    private Font font;
    private Font result;

    private boolean computed;
```

```
  private int first;
  private int second;
  private int tie;

  private int firstCards[];
private int secondCards[];

  private Random random;

  private int high_pair1;
  private int low_pair1;
private int three1;
  private int four1;
  private int high_pair2;
  private int low_pair2;
private int three2;
  private int four2;

  public void init() { ... }
  public void paint(Graphics g) { ... }
  private double getOneDigit(int x) { ... }
  private void drawCard(Graphics g,
            int x, int y, int selectedCard)
                                    { ... }

  public void actionPerformed(
                      ActionEvent e) { ... }

  public void mouseClicked(
                       MouseEvent e) { ... }
  public void mousePressed(MouseEvent e)
                                    { ... }
  public void mouseReleased(MouseEvent e)
                                    { ... }
  public void mouseEntered(MouseEvent e)
                                    { ... }
  public void mouseExited(MouseEvent e)
                                    { ... }
  private int getAction(Point p) { ... }

  private void reset() { ... }
  private void compute() { ... }
  private void generateCards() { ... }
  private int getUnusedCard() { ... }
  private void resetCards() { ... }
  private int findWinner() { ... }
  private void sortHands() { ... }

  private int computeStraigh(int[] cards)
                                    { ... }
  private int computeFlush(int[] cards)
                                    { ... }
  private int computeStraightFlush(
          int[] cards, int flush) { ... }
  private void computePairs1() { ... }
  private void computePairs2() { ... }
  private int getStraighFlushtWinner(
                        int straight_flush1,
                        int straight_flush2)
                                    { ... }
  private int getFourWinner(
                int four1, int four2) { ... }
  private int getFullHouseWinner() { ... }
  private int getFlushWinner(
                int flush1, int flush2) { ... }
  private int getStraightWinner(int
              straight1, int straight2) { ... }
  private int getThreeWinner(
                 int three1, int three2) { ... }
  private int getTwoPairsWinner() { ... }
  private int getOnePairWinner(int
          high_pair1, int high_pair2) { ... }
  private int getHighCardWinner(
          int noCards, int first, int second,
                                int suit) { ... }
}
```

Constante

Au fost definite mai multe constante pentru a mări lizibilitatea codului.

Prima dintre ele este MAX, care indică numărul configurațiilor care vor fi generate aleator în încercarea de a estima șansele de câștig ale jucătorilor.

Urmează patru constante care reprezintă cele patru suite: **HEARTS**, **SPADES**, **CLUBS** și **DIAMONDS**.

Pentru fiecare dintre acțiunile posibile am definit câte o constantă. Acțiunile posibile și constantele corespunzătoare sunt:

- schimbarea suitei curente (cea din care pot fi alese cărți) la cupe (CHANGE_TO_HEARTS);
- schimbarea suitei curente la trefle (CHANGE_TO_CLUBS);
- schimbarea suitei curente la pici (CHANGE_TO_SPADES);
- schimbarea suitei curente la carouri (CHANGE_TO_DIAMONDS);
- alegerea unei cărți (CARD_SELECTED);
- renunțarea la ultima carte aleasă (CARD_REMOVED);
- revenirea la configurația inițială (RESET);
- estimarea șanselor (COMPUTE);

Am mai definit constante pentru a identifica jucătorul care a câștigat într-o anumită situație: **FIRST** pentru primul jucător, **SECOND** pentru al doilea, respectiv **TIE** în caz de egalitate.

Variabile membre

Vom avea o variabilă mainSuit care va conține suită curentă (cea din care pot fi alese cărți). Această variabilă va avea ca valori una dintre cele patru constante pentru suite. Inițial valoarea acestei variabile va fi HEARTS (dacă nu schimbăm suita, la început pot fi alese cupe)

Avem un vector used care conține valori boolene; acesta va avea 52 de elemente și va indica dacă o anumită

carte este sau nu folosită deja în configurația curentă. O carte folosită nu va putea fi selectată din nou și nu va fi aleasă aleator în momentul în care se vor genera configurațiile pentru care vor fi stabilite rezultatele.

Vectorul de numere întregi `selectedCards` va conține codurile cărților deja selectate de utilizator. El va avea opt elemente, deoarece utilizatorul va putea alege cel mult opt cărți.

Numărul întreg `noSelected` va indica numărul cărților selectate.

Vectorul `values` conține codurile pentru valorile cărților; este utilizat pentru a identifica denumirile fișierelor care conțin imaginile cărților.

De asemenea, vectorul `colors` conține codurile pentru culori și este și el utilizat pentru a identifica denumirile fișierelor care conțin imaginile cărților.

Vectorul `cards` va conține imaginile corespunzătoare celor 52 de cărți.

Avem patru obiecte `Image` care vor conține imaginile corespunzătoare fundalului, precum și cele corespunzătoare celorlalte trei suite. Cele patru obiecte, `background`, `spades`, `diamonds` și `clubs`, vor conține imaginile corespunzătoare fișierelor **`background.jpg`**, **`spades.jpg`**, **`diamonds.jpg`** și **`clubs.jpg`**.

Pentru afișarea mesajelor avem trei obiecte `Font`: `symbol`, `font` și `result`.

Variabila booleană `computed` va indica dacă șansele au fost calculate sau nu (pentru a fi sau nu afișate rezultatele).

Variabilele `first`, `second` și `tie` vor indica numărul situațiilor generate în care a câștigat primul jucător, al doilea jucător, respectiv a fost egalitate.

Vectorii `firstCards` și `secondCards` vor conține cele șapte cărți ale fiecărui jucător, pentru o configurație generată aleator.

Obiectul `random` ne va permite să generăm numere pseudoaleatoare.

Variabilele `high_pair1`, `low_pair1`, `three1` și `four1` vor indica, pentru primul jucător, valoarea celei mai mari perechi, valoarea celei mai mici perechi, valoarea celor trei bucăți de același fel sau valoarea cărților care formează un careu. Dacă jucătorul nu are pereche mare, pereche mică, trei bucăți sau careu, valorile corespunzătoare sunt `-1`.

Variabilele `high_pair2`, `low_pair2`, `three2` și `four2` au semnificații similare, dar pentru al doilea jucător.

Inițializarea applet-ului

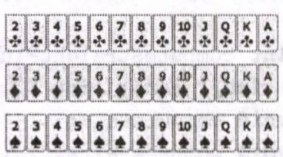
Metoda `init()` a *applet*-ului se va ocupa de crearea anumitor obiecte, setarea unor valori implicite și încărcarea unor imagini (pentru fundal și pentru suite; nu vor fi încărcate imaginile pentru cărți).

Codul sursă al acestei metode este prezentat în cele ce urmează:

```
public void init()
{
  mainSuit = HEARTS;
  try {
    background = Toolkit.getDefaultToolkit().
            getImage(new URL(getCodeBase() +
                             "background.jpg"));
    diamonds = Toolkit.getDefaultToolkit().
            getImage(new URL(getCodeBase() +
                               "diamonds.jpg"));
    clubs = Toolkit.getDefaultToolkit().
            getImage(new URL(getCodeBase() +
                                 "clubs.jpg"));
    spades = Toolkit.getDefaultToolkit().
            getImage(new URL(getCodeBase() +
                               "spades.jpg"));
  }
  catch (MalformedURLException e)
  {
    e.printStackTrace();
  }

  addMouseListener(this);

  used = new boolean[52];
  selectedCards = new int[8];
  firstCards = new int[7];
  secondCards = new int[7];

  noSelected = 0;

  reset();

  symbol = new Font("Timew New Roman", 0, 24);
  font = new Font("Tahoma", Font.BOLD, 18);
  result = new Font("Tahoma", Font.BOLD, 10);

  random = new Random();

  cards = new Image[52];

  for (int i = 0; i < 52; i++)
  {
    cards[i] = null;
  }
}
```

Desenarea ferestrei

În metoda `paint()` vom avea o desenare a fundalului; apoi, în cazul în care cupele nu reprezintă suita curentă, avem o desenare a imaginii care conține cărțile suitei curente (această imagine va fi desenată peste cupele din imaginea de fundal). Cele trei imagini care ar putea fi desenate în această zonă sunt:



Pentru suita curentă, vom verifica dacă există cărți deja folosite. Pentru fiecare carte folosită vom desena un dreptunghi alb în zona corespunzătoare.

În funcție de numărul cărților deja selectate, vom desena aceste cărți în zonele corespunzătoare (vom efectua apeluri ale metodei `drawCard`).

Vom colora un dreptunghi alb peste butonul prin care am putea cere evaluarea șanselor dacă această cerere nu poate fi satisfăcută acum. Butonul va fi "activ" doar înainte de flop (după alegerea tuturor cărților celor doi jucători), după flop sau după turn. Cu alte cuvinte, butonul nu va dispărea dacă vom avea patru, șapte sau opt cărți selectate și va dispărea în celelalte situații (una, două, trei, cinci sau șase cărți selectate).

În cazul în care au fost calculate șansele, acestea vor fi afișate.

Prezentăm în continuare codul sursă al acestei metode:

```
public void paint(Graphics g)
{
  g.setFont(font);
  g.drawImage(background, 0, 0, this);

  switch (mainSuit) {
    case HEARTS:
      break;
    case CLUBS:
      g.drawImage(clubs, 44, 136, this);
      break;
    case DIAMONDS:
      g.drawImage(diamonds, 44, 136, this);
      break;
    case SPADES:
      g.drawImage(spades, 44, 136, this);
      break;
  }

  g.setColor(Color.WHITE);
  for (int i = 0; i < 13; i++)
  {
    if (used[4 * i + mainSuit])
    {
      g.fillRect(44 + 21 * i, 136, 21, 37);
    }
  }

  // observati lipsa comenzilor break
  // in functie de numarul cartilor selectate
  // vom desena un numar de carti
  switch (noSelected) {
    case 8:
      drawCard(g, 339, 45, selectedCards[7]);
    case 7:
      drawCard(g, 286, 45, selectedCards[6]);
    case 6:
      drawCard(g, 243, 45, selectedCards[5]);
    case 5:
      drawCard(g, 200, 45, selectedCards[4]);
    case 4:
      drawCard(g, 147, 45, selectedCards[3]);
    case 3:
      drawCard(g, 104, 45, selectedCards[2]);
    case 2:
      drawCard(g, 51, 45, selectedCards[1]);
    case 1:
      drawCard(g, 8, 45, selectedCards[0]);
  }

  if (noSelected != 4 && noSelected != 7 &&
                                  noSelected != 8)
  {
    g.setColor(Color.WHITE);
    g.fillRect(317, 136, 64, 40);
  }

  if (computed)
  {
    g.setColor(Color.BLUE);
    g.setFont(result);
    g.drawString("Juc\u0103torul #1: ",
                                      8, 110);
    g.drawString(getOneDigit(first) + "%",
                                     85, 110);
    g.drawString("Egalitate: ",
                                      8, 120);
    g.drawString(getOneDigit(tie) + "%",
                                     85, 120);
    g.drawString("Juc\u0103torul #2: ",
                                      8, 130);
    g.drawString(getOneDigit(second) + "%",
                                     85, 130);
  }
}
```

Pentru a afișa rezultatele cu o precizie de o zecimală am folosit următoarea metodă:

```
private double getOneDigit(int x){
  return (double)Math.round(1000.0 * x / MAX)
                                        / 10.0;
}
```

Pentru denearea unei cărți vom folosi metoda `drawCard`. Aceasta are ca argumente coordonatele la care trebuie desenată cartea, precum și numărul de ordine al cărții; de asemenea, avem și obiectul `Graphics` cu ajutorul căruia vom desena.

Pentru a desena imaginea, vom verifica dacă acesta a fost sau nu încărcată și o vom încărca dacă este cazul.

După ce ne asigurăm că imaginea este încărcată, ea va fi desenată.

Implementarea este următoarea:

```
private void drawCard(Graphics g,
                      int x, int y, int selectedCard)
{
  g.setColor(Color.WHITE);
  g.setFont(font);

  if (cards[selectedCard] == null)
  {
    String name =
      values[selectedCard / 4].toLowerCase()
        + colors[selectedCard % 4] + ".jpg";
    try
    {
      cards[selectedCard] = Toolkit.
            getDefaultToolkit().getImage(new
                URL(getCodeBase() + name));
    }
    catch (MalformedURLException e)
    {
      e.printStackTrace();
    }
  }

  g.drawImage(cards[selectedCard], x, y - 4,
                                          this);
}
```

Efectuarea acțiunilor

În momentul efectuării unui *click* este posibil ca o anumită acțiune să fie necesară.

Detectarea acțiunilor care trebuie efectuate este implementată în metoda `getAction()`, care este apelată din interiorul metodei `mouseClicked()`.

Metoda `getAction()` are ca parametru punctul în care a fost efectuat un *click*. Pe baza coordonatelor acestui punct, metoda decide dacă trebuie efectuată sau nu o anumită acțiune.

Un click poate duce la schimbarea suitei curente, la selectarea unei cărți (dacă *click*-ul nu este în zona unei) cărți care a fost deja selectată, la eliminarea unei cărți (dacă *click*-ul este pe ultima carte selectată), resetarea configurației sau la estimarea șanselor (dacă butonul corespunzător este vizibil).

Implementarea acestei metode este prezentată în continuare:

```
private int getAction(Point p)
{
  int x = (int) p.getX();
  int y = (int) p.getY();

  if (x >= 8 && x <= 24 &&
                    y >= 136 && y <= 152)
  {
    return CHANGE_TO_HEARTS;
  }
  if (x >= 26 && x <= 42 &&
                    y >= 136 && y <= 152)
  {
    return CHANGE_TO_DIAMONDS;
  }
  if (x >= 8 && x <= 24 &&
                    y >= 154 && y <= 170)
  {
    return CHANGE_TO_CLUBS;
  }
  if (x >= 26 && x <= 42 &&
                    y >= 154 && y <= 170)
  {
    return CHANGE_TO_SPADES;
  }
  if (x >= 44 && x <= 316 &&
                    y >= 136 && y <= 170)
  {
    return CARD_SELECTED;
  }

  if (x >= 8 && x <= 47 &&
      y >= 41 && y <= 97 && noSelected == 1)
  {
    return CARD_REMOVED;
  }
  if (x >= 51 && x <= 90 &&
      y >= 41 && y <= 97 && noSelected == 2)
  {
    return CARD_REMOVED;
  }
  if (x >= 104 && x <= 143 &&
      y >= 41 && y <= 97 && noSelected == 3)
  {
    return CARD_REMOVED;
  }
  if (x >= 147 && x <= 186 &&
      y >= 41 && y <= 97 && noSelected == 4)
  {
    return CARD_REMOVED;
  }
  if (x >= 200 && x <= 239 &&
      y >= 41 && y <= 97 && noSelected == 5)
  {
    return CARD_REMOVED;
  }
  if (x >= 243 && x <= 282 &&
      y >= 41 && y <= 97 && noSelected == 6)
  {
    return CARD_REMOVED;
  }
  if (x >= 286 && x <= 325 &&
      y >= 41 && y <= 97 && noSelected == 7)
  {
    return CARD_REMOVED;
  }
```

```
  if (x >= 339 && x <= 378 &&
      y >= 41 && y <= 97 && noSelected == 8)
  {
    return CARD_REMOVED;
  }
  if (x >= 318 && x <= 378 &&
                      y >= 111 && y <= 128)
  {
    return RESET;
  }
  if (x >= 317 && x <= 377 &&
                      y >= 137 && y <= 173)
  {
    return COMPUTE;
  }
  return -1;
}
```

În funcție de acțiunea returnată de metoda `getAction()`, metoda `mouseClicked()` va efectua acțiuni sau va apela diferite alte metode care vor duce la executarea acestor acțiuni. Implementarea este:

```
public void mouseClicked(MouseEvent e) {
  Point p = e.getPoint();
  int action = getAction(p);
  switch (action)
  {
    case CHANGE_TO_HEARTS:
      mainSuit = HEARTS;
      repaint();
      break;
  case CHANGE_TO_DIAMONDS:
    mainSuit = DIAMONDS;
    repaint();
    break;
  case CHANGE_TO_SPADES:
    mainSuit = SPADES;
    repaint();
    break;
  case CHANGE_TO_CLUBS:
    mainSuit = CLUBS;
    repaint();
    break;
  case CARD_SELECTED:
    if (noSelected >= 8)
    {
      break;
    }
    // dimensiunea unei carti este de 21 de
    // pixeli
    int card = ((int) p.getX() - 44) / 21;
    if (!used[4 * card + mainSuit])
    {
      selectedCards[noSelected++] =
                          4 * card + mainSuit;
      used[4 * card + mainSuit] = true;
      computed = false;
      repaint();
    }
    break;
  case CARD_REMOVED:
    if (noSelected > 0)
    {
      noSelected--;
      used[selectedCards[noSelected]] =
                                    false;
      computed = false;
      repaint();
    }
    break;
  case RESET:
     if (noSelected > 0)
     {
       reset();
       repaint();
     }
     break;
  case COMPUTE:
    if (noSelected == 4 || noSelected == 7
                        || noSelected == 8)
    {
      compute();
      repaint();
    }
  }
}
```

Resetarea configurației

Metoda `reset()` este apelată la început și de fiecare dată când este apăsat butonul *Anulează*.

Prin apelul acestei metode numărul cărților selectate devine 0 și toate cărțile sunt marcate ca fiind nefolosite.

Implementarea este următoarea:

```
private void reset()
{
  noSelected = 0;
  for (int i = 0; i < 52; i++)
  {
    used[i] = false;
  }
  computed = false;
}
```

Estimarea șanselor

Estimarea propriu-zisă a șanselor celor doi jucători se realizează cu ajutorul metodei `compute()`.

Pentru început, numărul situațiilor în care câștigă primul jucător, numărul situațiilor în care câștigă al doilea jucător, precum și numărul situațiilor în care avem egalitate sunt inițializate cu 0.

În continuare sunt generate aleator cele MAX configurații posibile. Pentru fiecare astfel de configurație se stabilește jucătorul care câștigă mâna sau faptul că avem egalitate și se incrementează contorul corespunzător (first, second sau tie).

Ulterior, cărțile marcate ca fiind utilizate în timpul generării configurației sunt demarcate pentru a putea fi alese la configurația următoare.

Implementarea acestei metode este următoarea:

```
private void compute() {
  computed = true;
  first = second = tie = 0;
  for (int i = 0; i < MAX; i++)
  {
    generateCards();
    switch (findWinner())
    {
      case FIRST:
        first++;
        break;
      case SECOND:
        second++;
        break;
      case TIE:
        tie++;
        break;
    }
    resetCards();
  }
}
```

Pentru a genera o configurație folosim metoda generateCards(). Aceasta completează vectorii firstCards și secondCards astfel:

- cărțile alese pentru primul jucător devin primele două elemente ale vectorului firstCards;
- cărțile alese pentru al doilea jucător devin primele două elemente ale vectorului secondCards;
- eventualele cărți alese pentru flop și turn devin următoarele elemente, începând cu a treia poziție (indice 2), atât în vectorul firstCards, cât și în vectorul secondCards;
- pentru pozițiile pentru care nu avem cărți selectate (river cu siguranță, eventual flop și turn) alegem aleator o carte nefolosită până acum.

Prezentăm acum implementarea acestei metode:

```
private void generateCards()
{
  // primul jucator
  firstCards[0] = selectedCards[0];
  firstCards[1] = selectedCards[1];
  // al doilea jucator
  secondCards[0] = selectedCards[2];
  secondCards[1] = selectedCards[3];
  for (int i = 0; i < 5; i++)
  {
    if (noSelected >= i + 5)
    {
      firstCards[i + 2] =
                        selectedCards[i + 4];
      secondCards[i + 2] =
                        selectedCards[i + 4];
    }
    else
    {
      int x = getUnusedCard();
      firstCards[i + 2] = x;
      secondCards[i + 2] = x;
      used[x] = true;
    }
  }
}
```

Metoda getUnusedCard() furnizează aleator o carte care nu a fost aleasă până în acel moment, implementarea fiind foarte simplă:

```
private int getUnusedCard()
{
  int x;
  do
  {
    x = random.nextInt(52);
  }
  while (used[x]);
  return x;
}
```

Determinarea câștigătorului

Pentru o configurație generată aleator, vom apela metoda findWinner() care va determina câștigătorul (sau faptul că avem egalitate). Această metoda va returna una dintre constantele FIRST, SECOND sau TIE.

Pentru început este apelată metoda sortHands() care realizează o sortare în ordine crescătoare a celor șapte cărți pe care le au la dispoziție cei doi jucători (firstCards și secondCards):

```
private void sortHands()
{
  boolean sorted;
  do
  {
    sorted = true;
    for (int i = 0; i < 6; i++)
    {
      if (firstCards[i] > firstCards[i + 1])
      {
        sorted = false;
        firstCards[i] ^= firstCards[i + 1];
```

```
            firstCards[i + 1] =
              firstCards[i] ^ firstCards[i + 1];
            firstCards[i] ^= firstCards[i + 1];
          }
        }
      }
      while (!sorted);
      do {
        sorted = true;
        for (int i = 0; i < 6; i++)
        {
          if (secondCards[i] > secondCards[i + 1])
          {
            sorted = false;
            secondCards[i] ^= secondCards[i + 1];
            secondCards[i + 1] =
              secondCards[i] ^ secondCards[i + 1];
            secondCards[i] ^= secondCards[i + 1];
          }
        }
      }
      while (!sorted);
    }
```

După sortarea cărților se încearcă încadrarea mâinilor în una dintre categoriile menționate la începutul acestui articol.

Pentru început vom verifica dacă avem chintă și/sau culoare.

Avem variabilele locale `flush1`, `flush2`, `straight1`, `straigh2`, `straight_flush1` și `straight_flush2` care vor indica dacă fiecare dintre cei doi jucători are chintă, culoare sau chintă de culoare (valorile -1 vor indica faptul că nu există chintă, culoare sau chintă de culoare).

Variabilele `flush1` și `flush2`, dacă nu au valoarea -1, indică suita din care fac parte cele cinci cărți care formează culoarea.

Variabilele `straight1`, `straigh2`, `straight_flush1` și `straight_flush2`, dacă nu au valoarea -1, indică valoarea cea mai mare a cărții care face parte din chintă.

Valorile variabilelor `flush1` și `flush2` sunt determinate prin apelul metodei `computeFlush()`.

Valorile variabilelor `straight1` și `straight2` sunt determinate prin apelul metodei `computeStraight()`.

În cazul în care avem chintă și culoare, am putea avea chintă de culoare, dar acest lucru nu este obligatoriu. Prezentăm în continuare o situație în care avem, pentru ambii jucători, atât chintă, cât și culoare, dar pentru niciunul nu avem chintă de culoare.

Pentru a verifica dacă, într-adevăr, avem chintă de culoare, trebuie să ne asigurăm că avem chintă și culoare și apoi să apelăm metoda `computeStraightFlush()`.

Acum putem verifica dacă cel puțin unul dintre jucători are chintă de culoare și determina rezultatul mâinii prin apelul metodei `getStraightFlushWinner()`.

În continuare vom putea determina valorile variabilelor `high_pair1`, `low_pair1`, `three1` și `four1` prin apelul metodei `computePairs1()`.

De asemenea, valorile variabilelor `high_pair2`, `low_pair2`, `three2` și `four2` vor fi determinate prin apelul metodei `computePairs2()`.

Dacă cel puțin unul dintre cei doi jucători are un careu, rezultatul mâinii este determinat prin apelarea metodei `getFourWinner()`.

În continuare, dacă cel puțin unul dintre cei doi jucători are ful, rezultatul mâinii este determinat prin apelarea metodei `getFullHouseWinner()`.

Trecem la verificarea culorilor; dacă cel puțin unul dintre cei doi jucători are o culoare, rezultatul mâinii este determinat prin intermediul apelului metodei `getFlushWinner()`.

Urmează chintele; în cazul în care cel puțin un jucător are chintă, rezultatul este determinat prin apelarea metodei `getStraightWinner()`.

În cazul în care cel puțin unul dintre jucători are trei bucăți, rezultatul mâinii va fi determinat prin apelul metodei `getThreeWinner()`.

Dacă cel puțin unul dintre jucători are două perechi, metoda care furnizează rezultatul mâinii respective este `getTwoPairsWinner()`.

Dacă unul dintre jucători are o pereche, atunci vom apela metoda `getOnePairWinner()` pentru a determina câștigătorul.

În sfârșit, dacă nu am stabilit încă nimic (mâna nu face parte din niciuna dintre categoriile anterioare), vom apela metoda `getHighCardWinner()`.

Implementarea metodei `findWinner()` este prezentată în continuare:

```
private int findWinner()
{
  int flush1 = -1;
  int flush2 = -1;
  int straight1 = -1;
  int straight2 = -1;
  int straight_flush1 = -1;
  int straight_flush2 = -1;

  sortHands();

  flush1 = computeFlush(firstCards);
  flush2 = computeFlush(secondCards);

  straight1 = computeStraigh(firstCards);
  straight2 = computeStraigh(secondCards);
```

```
if (flush1 == -1 || straight1 == -1)
{
  straight_flush1 = -1;
}
else
{
  straight_flush1 = computeStraightFlush(
                        firstCards, flush1);
}

if (flush2 == -1 || straight2 == -1)
{
  straight_flush2 = -1;
}
else
{
  straight_flush2 = computeStraightFlush(
                        secondCards, flush2);
}

// verificare chinta de culoare
if (straight_flush1 != -1
                    || straight_flush2 != -1)
{
    return getStraightFlushWinner(
          straight_flush1, straight_flush2);
}

computePairs1();
computePairs2();

// verificare careu
if (four1 != -1 || four2 != -1)
{
  return getFourWinner(four1, four2);
}

// verificare ful
if ((three1 != -1 && high_pair1 != -1) ||
        (three2 != -1 && high_pair2 != -1))
{
  return getFullHouseWinner();
}

// verificare culoare
if (flush1 != -1 || flush2 != -1)
{
  return getFlushWinner(flush1, flush2);
}

// verificare chinta
if (straight1 != -1 || straight2 != -1)
{
  return getStraightWinner(straight1,
                                straight2);
}

// verificare trei bucati
if (three1 != -1 || three2 != -1)
{
  return getThreeWinner(three1, three2);
}

// verificare doua perechi
if ((low_pair1 != -1 && high_pair1 != -1) ||
     (low_pair2 != -1 && high_pair2 != -1))
{
    return getTwoPairsWinner();
}

// verificare o pereche
if (high_pair1 != -1 || high_pair2 != -1)
{
  return getOnePairWinner(high_pair1,
                                high_pair2);
}

// verificare carte mare
return getHighCardWinner(5, -1, -1, -1);
}
```

Determinarea unei culori

Un jucător are o culoare dacă cel puțin cinci dintre cele șapte cărți pe care le are la dispoziție fac parte din aceeași suită.

Metoda `computeFlush()` are ca parametru un vector care conține cele șapte cărți. Ea va număra cărțile din fiecare culoare (în vectorul local `flushes`) și, în cazul în care vom avea un element a cărui valoare este cel puțin 5, atunci va fi returnată culoarea corespunzătoare. Dacă nu avem culoare, valoarea returnată va fi -1.

Implementarea este următoarea:

```
private int computeFlush(int[] cards)
{
  int flushes[] = {0, 0, 0, 0};
  for (int i = 0; i < 7; i++)
  {
    flushes[cards[i] % 4]++;
  }
  for (int i = 0; i < 4; i++)
  {
    if (flushes[i] >= 5)
    {
      return i;
    }
  }
  return -1;
}
```

Determinarea unei chinte

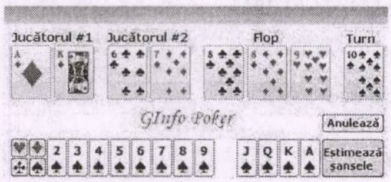
Un jucător are o chintă dacă cel puțin cinci dintre cele șapte cărți pe care le are la dispoziție au valori consecutive.

Metoda `computeStraight()` are ca parametru un vector care conține cele șapte cărți. Vectorul este parcurs de la sfârșit spre început, începând cu penultima poziție.

Folosim variabila locală `count` pentru a contoriza numărul cărților cu valori consecutive; aceasta este inițializată cu 0.

Dacă pe poziția curentă avem o carte cu o valoare cu 1 mai mică decât valoarea cărții de pe poziția următoare, atunci numărul cărților cu valori consecutive este incrementat.

Dacă pe poziția curentă avem o carte care are aceeași valoarea ca și cartea de pe poziția următoare, atunci numărul cărților rămâne neschimbat (putem avea chintă chiar dacă avem și cărți cu valori egale, așa cum se arată în imaginea de mai jos).



Dacă diferența dintre valoarea cărții curente și cea a cărții următoare este de cel puțin 2, atunci contorul va fi resetat la 0.

Dacă valoarea contorului ajunge la 4, înseamnă că avem cinci cărți cu valori consecutive. Returnăm valoarea celei mai mari cărți din chintă (adăugăm 4 la valoarea cărții curente).

Dacă am ajuns la prima carte, aceasta este un doiar și avem patru cărți consecutive, atunci am putea avea o chintă dacă ultima carte este un as. În acest caz returnăm valoarea 3 (chintă la cincear).

Trebuie observat faptul că, datorită faptului că valorile cărților sunt cuprinse între 0 și 12, vom verifica dacă prima carte are valoarea 0 (corespunzător doiarului) și dacă ultima carte are valoarea 12 (corespunzător asului); de asemenea returnăm valoarea 3 (corespunzător cincearului).

Dacă nu am identificat cinci cărți cu valori consecutive, returnăm valoarea -1, care indică faptul că nu avem nicio chintă.

Datorită faptului că parcurgem vectorul de la dreapta la stânga, vom identifica mai întâi chinta cea mai mare. Așadar, chiar dacă avem șase sau șapte cărți consecutive, vom returna corect valoarea maximă a unei cărți din chintă.

Implementarea este prezentată în continuare:

```
private int computeStraigh(int[] cards)
{
  int count = 0;
  for (int i = 5; i >= 0; i--)
  {
    if (cards[i] / 4 == cards[i + 1] / 4 - 1)
    {
      count++;
    }
    else
    {
      if (cards[i] / 4 != cards[i + 1] / 4)
      {
        count = 0;
      }
    }

    if (count == 4)
    {
      return cards[i] / 4 + 4;
    }

    if (cards[0] / 4 == 0 && cards[6] / 4 == 12
                                  && count == 3)
    {
      return 3;
    }

    return -1;
}
```

Determinarea chintei de culoare

Dacă avem chintă și culoare, putem verifica dacă avem chintă de culoare cu ajutorul metodei `computeStraightFlush()`. Aceasta are ca argumente un vector care conține cele șapte cărți, precum și culoarea pentru care se realizează verificarea.

Implementarea este similară celei pentru verificarea chintei dar, de această dată, va trebui să ignorăm cărțile care nu au culoarea corectă.

Vom crea un vector local `flush_cards` care conține cărțile de culoarea dată și vom aplica aceiași pași ca și pentru chinta simplă, dar folosind doar vectorul local.

Pentru a simplifica puțin codul, vectorul `flush_cards` nu va conține coduri ale cărților (cuprinse între 0 și 51), ci doar valori (cuprinse între 0 și 12).

Datorită faptului că lucrăm cu o singură culoare, nu putem avea cărți cu valori egale.

Implementarea este următoarea:

```
private int computeStraightFlush(int[] cards,
                                 int flush)
{
  int flush_cards[] = new int[7];
  int noFlushCards = 0;
  for (int i = 0; i < 7; i++)
  {
    if (cards[i] % 4 == flush)
    {
      flush_cards[noFlushCards++] =
                                 cards[i] / 4;
    }
  }
  int count = 0;
```

```
for (int i = noFlushCards - 2; i >= 0; i--)
{
  if (flush_cards[i] ==
                    flush_cards[i + 1] - 1)
  {
      count++;
  }
  else
  {
    count = 0;
  }

  if (count == 4)
  {
    return flush_cards[i] + 4;
  }

if (flush_cards[0] == 0 &&
        flush_cards[noFlushCards - 1] == 12
                              && count == 3)
  {
    return 3;
  }
return -1;
}
```

Perechi, trei bucăți, careu

Cu ajutorul metodelor `computePairs1()` și `computePairs2()` vom determina valorile variabilelor `high_pair1`, `low_pair1`, `three1`, `four1`, `high_pair2`, `low_pair2`, `three2` și `four2`, care sunt folosite pentru determinarea câștigătorului.

Reamintim faptul că variabilele `high_pair1`, `low_pair1`, `three1` și `four1` vor indica, pentru primul jucător, valoarea celei mai mari perechi, valoarea celei mai mici perechi, valoarea celor trei bucăți de același fel sau valoarea cărților care formează un careu. Dacă jucătorul nu are pereche mare, pereche mică, trei bucăți sau careu, valorile corespunzătoare sunt -1. De asemenea, variabilele `high_pair2`, `low_pair2`, `three2` și `four2` au semnificații similare, dar pentru al doilea jucător.

Pentru început, toate cele patru variabile, corespunzătoare unui jucător, sunt inițializate cu valoarea -1.

Folosim un contor `count` pentru a număra cărțile cu aceeași valoare; acesta este inițializat cu 1.

La fel ca în cazul chintei, parcurgem șirul cărților de la stânga spre dreapta, începând cu penultima poziție.

Dacă pe poziția curentă avem o carte cu aceeași valoare ca și cartea de pe poziția următoare, contorul este incrementat. Dacă valoarea este diferită, înseamnă că am terminat numărarea cărților cu o anumită valoare și apelăm metoda `checkPair1()`, respectiv metoda `checkPair2()`, pentru a actualiza, dacă este cazul, cele patru variabile.

Parcurgerea de la dreapta la stânga ne va asigura de faptul că obținem mai întâi perechile mai mari (eventual trei bucăți cu valoarea mai mare).

Metodele `checkPair1()` și `checkPair2()` sunt apelate încă o dată după terminarea parcurgerii, pentru ultima poziție parcursă (prima carte),

Implementările celor două metode sunt:

```
private void computePairs1() {
  high_pair1 = -1;
  low_pair1 = -1;
  three1 = -1;
  four1 = -1;

  int count = 1;

  for (int i = 5; i >= 0; i--)
  {
    if (firstCards[i] / 4 ==
                        firstCards[i + 1] / 4)
    {
      count++;
    }
    else
    {
      checkPair1(firstCards[i + 1] / 4,
                                      count);
      count = 1;
    }
  }

  checkPair1(firstCards[0] / 4, count);
}

private void computePairs2() {
  high_pair2 = -1;
  low_pair2 = -1;
  three2 = -1;
  four2 = -1;

  int count = 1;

  for (int i = 5; i >= 0; i--)
  {
    if (secondCards[i] / 4 ==
                      secondCards[i + 1] / 4)
    {
      count++;
    }
    else
    {
      checkPair2(secondCards[i + 1] / 4,
                                      count);

      count = 1;
    }
  }

  checkPair2(secondCards[0] / 4, count);
}
```

Actualizarea propriu-zisă a variabilelor high_pair1, low_pair1, three1 și four1, respectiv high_pair2, low_pair2, three2 și four2 are loc în interiorul metodelor checkPair1(), respectiv checkPair2().

Acestea au ca parametri valoarea contorului în momentul în care au fost apelate din interiorul metodelor computePairs1(), respectiv computePairs2(), precum și valoarea cărții curente din momentul apelului.

Practic argumentele indică numărul cărților cu o anumită valoare care sunt procesate.

Denumirile metodelor ar putea fi oarecum confuze, deoarece nu operează numai asupra perechilor ci și asupra a trei sau patru cărți cu aceeași valoare.

Totuși, în principal, aceste metode sunt apelate pentru perechi.

În funcție de valoarea contorului, se vor seta anumite variabile.

Dacă valoarea contorului este 4, atunci avem un careu, deci setăm valorile four1, respectiv four2.

Dacă valoarea contorului este 3, atunci avem trei bucăți și setăm valorile three1, respectiv three2, dar numai dacă nu au fost deja setate anterior. Dacă au fost setate, înseamnă că avem deja trei bucăți cu valori mai mari și avem un ful, deci valoarea curentă va reprezenta o pereche.

Dacă valoarea contorului este 2, atunci avem o pereche. Verificăm dacă perechea mare este deja setată (valorile variabilelor high_pair1, respectiv high_pair2, sunt diferite de -1) și o setăm dacă este cazul.

Dacă perechea mare era setată, atunci verificăm perechea mică (valorile variabilelor low_pair1, respectiv low_pair2, sunt diferite de -1) și o setăm dacă este cazul.

Există posibilitatea ca ambele perechi să fie deja setate (avem trei perechi în cele șapte cărți disponibile), caz în care nu setăm nicio variabilă.

Dacă valoarea contorului este 1, atunci nu setăm nicio variabilă. Contorul nu poate avea nicio altă valoare.

Implementările celor două metode sunt prezentate în continuare:

```
private void checkPair1(int val, int count)
{
  switch (count)
  {
    case 4:
      four1 = val;
      return;
    case 3:
      if (three1 == -1)
      {
        three1 = val;
      }
      else
      {
        if (val > high_pair1)
        {
          low_pair1 = high_pair1;
          high_pair1 = val;
        }
        else
        {
          if (val > low_pair1)
          {
            low_pair1 = val;
          }
        }
      }
      return;
    case 2:
      if (high_pair1 == -1)
      {
        high_pair1 = val;
      }
      else
      {
        if (low_pair1 == -1)
        {
          low_pair1 = val;
        }
      }
      return;
    default:
      return;
  }
}

private void checkPair2(int val, int count)
{
  switch (count)
  {
    case 4:
      four2 = val;
      return;
    case 3:
      if (three2 == -1)
      {
        three2 = val;
      }
      else
      {
        if (val > high_pair2)
        {
          low_pair2 = high_pair2;
          high_pair2 = val;
        }
        else
        {
          if (val > low_pair2)
          {
            low_pair2 = val;
          }
        }
      }
```

```
      return;
    case 2:
      if (high_pair2 == -1)
      {
        high_pair1 = val;
      }
      else
      {
        if (low_pair1 == -1)
        {
          low_pair1 = val;
        }
      }
      return;
    default:
      return;
  }
}
```

Câștigătorul pentru chinta de culoare

Metoda `getStraightFlushWinner()` este apelată în cazul în care cel puțin unul dintre jucători are o chintă de culoare. Ea are ca argumente valorile maxime ale cărților din chintele de culoare ale celor doi jucători. Aceste argumente au valoarea -1 dacă unul dintre jucători nu are chintă de culoare.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

Implementarea acestei metode este:

```
private int getStraightFlushWinner(
    int straight_flush1, int straight_flush2)
{
  if (straight_flush1 > straight_flush2)
  {
    return FIRST;
  }
  else
  {
    if (straight_flush1 < straight_flush2)
    {
      return SECOND;
    }
    else
    {
      return TIE;
    }
  }
}
```

Câștigătorul pentru careu

Metoda `getFourWinner()` este apelată în cazul în care cel puțin unul dintre jucători are un careu. Ea are ca argumente valorile cărților din careurile celor doi jucători. Aceste argumente au valoarea -1 dacă unul dintre jucători nu are careu.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

În cazul în care cei doi jucători au același careu, se apelează metoda `getHighCardWinner()` pentru o carte de orice culoare, cu orice valoare diferită de cea a cărților din careu, deoarece câștigătorul va fi dat de cea de-a cincea carte.

Implementarea acestei metode este:

```
private int getFourWinner(
                          int four1, int four2)
{
  if (four1 > four2)
  {
    return FIRST;
  }
  else
  {
    if (four1 < four2)
    {
      return SECOND;
    }
    else
    {
      return getHighCardWinner(1, four1,
                                -1, -1);
    }
  }
}
```

Câștigătorul pentru ful

Metoda `getFullHouseWinner()` este apelată în cazul în care cel puțin unul dintre jucători are un ful.

Aceasta operează cu variabilele `three1`, `three2`, `high_pair1` și `high_pair2`. Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

Inițial se verifică dacă ambii jucători au ful. Dacă unul dintre ei nu are ful, celălalt va fi câștigător.

Dacă ambii jucători au ful, atunci se verifică valoarea celor trei bucăți și, dacă aceste valori sunt egale, valorile perechii. Dacă și perechile sunt egale, atunci avem egalitate.

Implementarea acestei metode este:

```
private int getFullHouseWinner()
{
  if (three1 == -1 || high_pair1 == -1)
  {
    return SECOND;
  }
  if (three2 == -1 || high_pair2 == -1)
  {
    return FIRST;
  }
  if (three1 > three2)
  {
    return FIRST;
  }
```

```
    else
    {
      if (three2 > three1)
      {
        return SECOND;
      }
      else
      {
        if (high_pair1 > high_pair2)
        {
          return FIRST;
        }
        else
        {
          if (high_pair2 > high_pair1)
          {
            return SECOND;
          }
          else
          {
            return TIE;
          }
        }
      }
    }
}
```

Câștigătorul pentru culoare

Metoda `getFlushWinner()` este apelată în cazul în care cel puțin unul dintre cei doi jucători are o culoare. Ea are ca argumente suitele corespunzătoare culorilor celor doi jucători. În jocul *hold'em* poker jucătorii nu pot avea culori diferite.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

Inițial se verifică dacă ambii jucători au culoare. Dacă unul dintre jucători nu are culoare, atunci celălalt este declarat câștigător.

Dacă ambii jucători au culoare, atunci este apelată metoda `getHighCardWinner()` pentru cinci cărți de culoarea dată de argumente.

Implementarea acestei metode este:

```
private int getFlushWinner(int flush1, int
                                        flush2)
{
  if (flush1 == -1)
  {
    return SECOND;
  }
  else
  {
    if (flush2 == -1)
    {
      return FIRST;
    }
    else
    {
      return getHighCardWinner(5, -1, -1,
                                        flush1);
    }
  }
}
```

Câștigătorul pentru chintă

Metoda `getStraightWinner()` este apelată în cazul în care cel puțin unul dintre jucători are o chintă. Ea are ca argumente valorile maxime ale cărților din chintele celor doi jucători. Aceste argumente au valoarea -1 dacă unul dintre jucători nu are chintă.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`; ea este foarte asemănătoare cu metoda `getStraightFlushWinner()` (practic, implementarea celor două metode este identică; avem două funcții doar din motive de lizibilitate).

Implementarea acestei metode este:

```
private int getStraightWinner(
                int straight1, int straight2)
{
  if (straight1 > straight2)
  {
    return FIRST;
  }
  else
  {
    if (straight1 < straight2)
    {
      return SECOND;
    }
    else
    {
      return TIE;
    }
  }
}
```

Câștigătorul pentru trei bucăți

Metoda `getThreeWinner()` este apelată în cazul în care cel puțin unul dintre jucători are trei bucăți. Ea are ca argumente valorile cărților din cele trei bucăți ale celor doi jucători. Aceste argumente au valoarea -1 dacă unul dintre jucători nu are trei bucăți.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

În cazul în care cei doi jucători au aceleași trei bucăți, se apelează metoda `getHighCardWinner()` pentru două cărți de orice culoare, cu orice valoare diferită de cea a celor trei bucăți, deoarece câștigătorul va fi dat de cea de-a patra sau de cea de-a cincea carte (în caz de egalitate pentru a patra).

Implementarea acestei metode este:

```
private int getThreeWinner(
                        int three1, int three2)
{
  if (three1 > three2)
  {
    return FIRST;
  }
  else
  {
    if (three1 < three2)
    {
      return SECOND;
    }
    else
    {
      return getHighCardWinner(2, three1,
                                    -1, -1);
    }
  }
}
```

Câștigătorul pentru două perechi

Metoda `getTwoPairsWinner()` este apelată în cazul în care cel puțin unul dintre jucători are două perechi.

Aceasta operează cu variabilele `low_pair1`, `low_pair2`, `high_pair1` și `high_pair2`.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

Inițial se verifică dacă ambii jucători au două perechi. Dacă unul dintre ei nu are două perechi, celălalt jucător va fi câștigător.

Dacă ambii jucător au două perechi, atunci se verifică valorile perechii mari; dacă aceste valori sunt egale, se verifică valorile perechii mici.

Dacă și perechile mici sunt egale, atunci se apelează metoda `getHighCardWinner()` pentru o carte de orice culoare, cu orice valoare diferită de cele a cărților care formează perechile, deoarece câștigătorul va fi dat de cea de-a cincea carte.

Implementarea acestei metode este:

```
private int getTwoPairsWinner()
{
  if (high_pair1 == -1 || low_pair1 == -1)
  {
    return SECOND;
  }
  if (high_pair2 == -1 || low_pair2 == -1)
  {
    return FIRST;
  }
  if (high_pair1 > high_pair2)
  {
    return FIRST;
  }
  else
  {
    if (high_pair2 > high_pair1)
    {
      return SECOND;
    }
    else
    {
      if (low_pair1 > low_pair2)
      {
        return FIRST;
      }
      else
      {
        if (low_pair2 > low_pair1)
        {
          return SECOND;
        }
        else
        {
          return getHighCardWinner(1,
                high_pair1, high_pair2, -1);
        }
      }
    }
  }
}
```

Câștigătorul pentru o pereche

Metoda `getOnePairWinner()` este apelată în cazul în care cel puțin unul dintre jucători are o pereche. Ea are ca argumente valorile cărților care formează perechile celor doi jucători. Aceste argumente au valoarea -1 dacă unul dintre jucători nu are o pereche.

Metoda returnează una dintre constantele `FIRST`, `SECOND` și `TIE`.

În cazul în care cei doi jucători au aceeași pereche, se apelează metoda `getHighCardWinner()` pentru trei cărți de orice culoare, cu orice valoare diferită de cea a cărților din pereche, deoarece câștigătorul va fi dat de cea de-a treia, cea de-a patra sau de cea de-a cincea carte.

Implementarea acestei metode este:

```
private int getOnePairWinner(
              int high_pair1, int high_pair2)
{
  if (high_pair1 > high_pair2)
  {
    return FIRST;
  }
  else
  {
    if (high_pair1 < high_pair2)
    {
      return SECOND;
    }
```

```
    else
    {
      return getHighCardWinner(3, high_pair1,
                                    -1, -1);
    }
  }
}
```

Departajarea pe baza celei mai mari cărți

În mai multe situații avem nevoie de departajarea pe baza celei mai mari cărți:

- niciuna dintre cele două mâini nu se încadrează în niciuna dintre categoriile amintite (pereche, două perechi, trei bucăți, chintă, culoare, ful, careu, chintă de culoare);
- ambii jucători au câte o pereche cu aceeași valoare;
- ambii jucători au câte două perechi cu aceeași valoare;
- ambii jucători au câte trei bucăți cu aceeași valoare;
- ambii jucători au culoare;
- ambii jucători au câte un careu cu aceeași valoare.

În funcție de situație trebuie să luăm în considerare una, două, trei sau toate cele cinci cărți. Mai mult pentru culoare trebuie să luăm în considerare doar cărțile care fac parte dintr-o anumită suită.

Pentru a avea o singură funcție am creat metoda `getHighCardWinner()` ai cărei parametri indică numărul cărților care vor fi luate în considerare, cele maximum două valori care trebuie ignorate, precum și culoarea cărților care trebuie considerate.

Valorile ignorate sau culoarea pot să nu fie precizate (argumentele corespunzătoare au valoarea -1).

Parametrul `noCards` indică numărul cărților care trebuie luate în considerare. Inițial vom crea doi vectori care conțin cărțile considerate pentru fiecare dintre jucători, parcurgând cărțile jucătorilor de la dreapta la stânga, începând cu ultima poziție și verificând dacă valoarea și culoarea cărții indică faptul că acea carte trebuie luată în considerare.

După crearea vectorilor (aceștia sunt ordonați crescător), sunt comparate elementele acestora până în momentul în care apare prima diferență.

Dacă după `noCards` comparații nu avem nicio diferență, atunci avem egalitate.

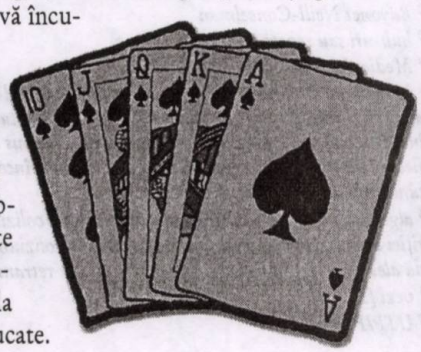
Implementarea acestei metode este:

```
private int getHighCardWinner(int noCards,
            int first, int second, int suit)
{
  int high_cards1[] = new int[7];
 int high_cards2[] = new int[7];
 int count1 = 0;
 int count2 = 0;

  for (int i = 0; i < 7; i++)
  {
    int card1 = firstCards[i] / 4;
    if (card1 != first && card1 != second &&
        (suit == -1 || suit == firstCards[i]
                                    % 4))
    {
      high_cards1[count1++] = card1;
    }
    int card2 = secondCards[i] / 4;
    if (card2 != first && card2 != second &&
     (suit == -1 || suit == secondCards[i] %
                                        4))
    {
        high_cards2[count2++] = card2;
    }
  }

  for (int i = 0; i < noCards; i++)
  {
    if (high_cards1[count1 - i - 1] >
                high_cards2[count2 - i - 1])
    {
      return FIRST;
    }
    else
    {
      if (high_cards1[count1 - i - 1] <
                high_cards2[count2 - i - 1])
      {
        return SECOND;
      }
    }
  }
  return TIE;
}
```

Concluzii

Acesta a fost un nou (lung) articol prin care încercăm popularizarea limbajului *Java*. În *C++* sau *Pascal* crearea unei astfel de aplicații ar fi fost mult mai complicată (în special partea grafică).

Nu am avut deloc intenția de a populariza jocurile de noroc. De fapt, *poker*-ul în versiunea *hold'em* nu este considerat a fi un joc de noroc pe termen lung. Evident, pentru o mână norocul are un cuvânt important dar, cunoașterea șanselor de câștig în diferite situații poate duce la obținerea unor strategii (nu tocmai sigure) de câștig.



Nu dorim să vă încurajăm să jucați *poker* dar, dacă o faceți (dacă aveți peste 18 ani, conform legii), vă urăm succes în obținerea unei chinte royale. Chinta royală apare odată la 30940 de mâini jucate.

MAC și ARP (I)

Tiberiu Socaciu

Domnul Sistemescu, prietenul nostru, are acum de lucru: învață. Are niscaiva probleme în rețea și a auzit că, dacă citește despre MAC-uri și ARP-uri, le poate rezolva singur, chiar dacă pe calculatoarele din rețea sunt filtre de tip firewall activate.

Coliziunea CDMA/CD[1]. Conexiunile BNC[2] și Hub[3]-urile

Unultre dintre primele medii de transmisie în rețea folosite a fost cablul coaxial. Datorită faptului că, potențial, mediul de transmisie trebuia partajat de mai mulți doritori care vor să efectueze transmisie de date, a trebuit ca fiecare "jucător" sa aibă o adresă unică, atribuită de producător.

Această adresă se numește *MAC*[4] și, cu ajutorul ei, se vor "semna" toate pachetele care ies prin respectiva interfață, respectiv, pe baza ei, se va decide dacă un pachet este sau nu adresat unei anumite interfețe.

Tehnica de transmitere a datelor *Ethernet* se numește *CDMA/CD*; ea permite detectarea faptului dacă două transmisii de date s-au suprapus în timp[5], generând o coliziune[6].

Acest principiu stă la baza primelor rețele *Ethernet*, conectate folosind conectori "T" la un cablu de rețea de tip *BNC*.

Datorită existenței unei limite pentru dimensiunea maximă a tronsoanelor *BNC*[7] care pot fi conectate, dar și a numărului maxim de tronsoane, au fost inventate repetoarele de semnal și *hub*-urile.

Principial, *hub*-ul rupe arhitectura liniară oferind conectivitate prin cabluri torsadate[8] la echipamentul conectat direct pe *BNC*[9]. Indiferent dacă un echipament era conectat via un port *TP* sau *BNC* la cablul coaxial, principiul de funcționare *CDMA/CD* a rămas și s-a menținut.

Important este de reținut faptul că, la rețelele *Ethernet*, este posibil ca un pachet[10] să fie primit și de alt destinatar[11] decât cel căruia îi este adresat[12]. De aceea, spunem că toate pachetele transmise în rețea au regim de *broadcast*.

Switch-uri și hub-uri. Comutarea de pachete[13]

Odată cu dispariția treptată a cablărilor tip *BNC* și apariția noilor "cablări structurate" folosind cabluri *TP*[14], datorită noilor echipamente de tip *hub Ethernet*, a început studierea posibilității creșterii vitezei de transmisie pe cabluri torsadate folosind tehnologii compatibile *Ethernet*.

Așa s-a ajuns la concluzia că s-ar putea crește viteza de transmisie de la 10 Mbps[15] la 100Mbps[16] doar prin modificarea tipului de torsadare[17], deoarece transmisiile de date foloseau câte o pereche pentru transmisie de date (*TX*), respectiv recepție de date (*RX*). Noul standard a fost numit *FastEthernet*.

Următorul pas a fost atașarea unei memorii pentru dispozitivele de tip *hub*, în care să se memoreze toate *MAC*-urile potențiale pentru destinațiile pachetelor.

---

[1] *Carrier Sense Multiple Access with Collision Detection*
[2] *Bayonet Neill-Concelman*
[3] *hub-uri sau repetoare*
[4] *Media Address Controller*
[5] *în mod normal, fiecare interfață de rețea trimite datele doar după ce se asigură că este "liniște" pe mediul de transmisie; totuși, pot apărea cazuri în care două sau mai multe interfețe au de transmis ceva cvasisimultan: fiecare dintre ele se asigură că linia este liberă și încep să transmită, putând apărea suprapunerea transmisiilor*
[6] *algoritmii CDMA/CD reușesc să identifice o coliziune și, în cazul apariției acesteia, fiecare membru participant la coliziune așteaptă o perioadă aleatoare de timp, după care pachetul este retransmis*
[7] *vezi [3]*
[8] *UTP/FTP/STP*
[9] *odată cu dispariția treptată a cablărilor BNC, au fost definite și hub-uri Ethernet fără conectori BNC*
[10] *pachetul conține un câmp obligatoriu în care se află MAC-ul expeditorului și un câmp opțional (completat dacă este posibil) în care se află MAC-ul destinatarului, alături de alte informații de structură și de datele propriu-zise*
[11] *din cauza topologiei liniare inițiale, care a impus tehnologia*
[12] *totuși, identificarea destinatarului se realizează pe baza MAC-ului cu care se completează câmpul corespunzător din pachet.*
[13] *switching*
[14] *Twisted Pair (cabluri torsadate)*
[15] *valoarea maximă pentru Ethernet*
[16] *valoarea maximă pentru FastEthernet*
[17] *astfel, pentru 10 Mbps ar fi suficient un cablu minim CAT3, iar pentru 100 Mbps un cablu minim CAT5*

Comanda nmap

```
[root@server ~]# nmap -sP 86.104.31.64/28

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2007-01-21 09:04 EET
Host 86.104.31.74 appears to be up.
MAC Address: 00:40:F4:B3:8C:96 (Cameo Communications)
Host 86.104.31.76 appears to be up.
MAC Address: 00:14:BF:3C:48:FD (Cisco-Linksys)
Host 86.104.31.77 appears to be up.
MAC Address: 00:30:05:29:60:9A (Fujitsu Siemens Computers)
Host 86.104.31.78 appears to be up.
Nmap finished: 16 IP addresses (4 hosts up) scanned in 1.303 seconds
```

Acest lucru se poate realiza printr-o banală analiză a expeditorului unui pachet care vine pe un port oarecare. Astfel, echipamentul ar putea "învăța" din mers *MAC*-uri, construind și întreținând în timp real o tabelă cu intrări de forma (port, *MAC*).

Probabil, cea mai bună organizare a tabelei este decisă de fabricantul echipamentului[18] în momentul în care scrie *firmware*-ul echipamentului. Utilitatea acestei tabele iese la iveală în momentul în care *software*-ul echipamentului decide ca la primirea unui pachet, acesta să fie transmis, pe cât posibil, doar pe portul pe care este prezumtiv conectată[19] interfața în cauză. Acest lucru generează scăderi drastice ale coliziunilor de pachete în noile echipamente, care se numesc *switch*-uri.

Cum putem afla tipul echipamentului pe baza adresei MAC?

Există baze de date cu producătorii de *hardware* și adresele *MAC* utilizate de aceștia în *controller*-ele de interfețe. De exemplu, sub *Linux*, avem o astfel de bază de date im-

[18] *arbori binari de căutare, hash-uri etc.*

[19] *direct sau indirect, prin alte hub-uri / switch-uri*

Exemplu IPCONFIG

```
C:\Documents and Settings\tibi>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : tibi
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Unknown
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Intel(R) PRO/100 VM Network Connection
        Physical Address. . . . . . . . . : 00-30-05-29-60-9A
        Dhcp Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 86.104.31.77
        Subnet Mask . . . . . . . . . . . : 255.255.255.240
        Default Gateway . . . . . . . . . : 86.104.31.78
        DNS Servers . . . . . . . . . . . : 193.230.240.16
                                            193.230.240.21
                                            193.230.240.22
                                            86.104.31.78
                                            86.104.31.70
        NetBIOS over Tcpip. . . . . . . . : Disabled
```

Comanda IPCONFIG

```
C:\Documents and Settings\tibi>ipconfig /?

USAGE:
    ipconfig [/? | /all | /renew [adapter] | /release [adapter] |
              /flushdns | /displaydns | /registerdns |
              /showclassid adapter |
              /setclassid adapter [classid] ]

where
    adapter             Connection name
                        (wildcard characters * and ? allowed, see examples)

    Options:
        /?              Display this help message
        /all            Display full configuration information.
        /release        Release the IP address for the specified adapter.
        /renew          Renew the IP address for the specified adapter.
        /flushdns       Purges the DNS Resolver cache.
        /registerdns    Refreshes all DHCP leases and re-registers DNS names
        /displaydns     Display the contents of the DNS Resolver Cache.
        /showclassid    Displays all the dhcp class IDs allowed for adapter.
        /setclassid     Modifies the dhcp class id.

The default is to display only the IP address, subnet mask and
default gateway for each adapter bound to TCP/IP.

For Release and Renew, if no adapter name is specified, then the IP address
leases for all adapters bound to TCP/IP will be released or renewed.

For Setclassid, if no ClassId is specified, then the ClassId is removed.

Examples:
    > ipconfig                  ... Show information.
    > ipconfig /all             ... Show detailed information
    > ipconfig /renew           ... renew all adapters
    > ipconfig /renew EL*       ... renew any connection that has its
                                    name starting with EL
    > ipconfig /release *Con*   ... release all matching connections,
                                    eg. "Local Area Connection 1" or
                                        "Local Area Connection 2"
```

plementată în programul nmap. În exemplul din casetă am cerut să se verifice rețeaua locală[20] și am aflat că în rețea utilizăm un echipament *wireless* cu chipset Cameo[21], avem un router *Linksys*[22] și o placă de rețea *Fujitsu Siemens*[23].

Astfel, putem afla și ce tipuri de echipamente posedă colegii noștri de rețea, diferiți clienți ai *ISP*-ului nostru, dar și ce echipament folosește stimabilul nostru furnizor de servicii.

Aflarea adresei MAC

De regulă, pe fiecare echipament care are interfețe cu adrese *MAC* înscrise în *controller*-ul interfețelor are scris pe ambalaj și/sau echipament această adresă. Urmăriți toate echipamentele la care aveți acces[24] și veți vedea etichete autocolante lipite pe carcasă și cutie, cu aceleași informații[25]. Totusi, în lipsa acestor informații, putem afla *MAC*-ul folosind comenzi ale sistemului de operare.

[20] *mai exact, subclasa 86.104.31.64/28*
[21] *un access point Trendnet*
[22] *un router Cisco Linksys WRT 54G*
[23] *un calculator Fujitsu Siemens cu placă de rețea onboard*

[24] *plăci de rețea, modem-uri de cablu, modem-uri ADSL, echipamente wireless, router-e etc.*
[25] *aceste informații se referă la MAC-ul original, cel înscris de producătorul controller-ului*

Comanda ifcomfig

```
[root@server ~]# ifconfig --help
Usage:
  ifconfig [-a] [-v] [-s] <interface> [[<AF>] <address>]
  [add <address>[/<prefixlen>]]
  [del <address>[/<prefixlen>]]
  [[-]broadcast [<address>]]  [[-]pointopoint [<address>]]
  [netmask <address>]  [dstaddr <address>]  [tunnel <address>]
  [outfill <NN>] [keepalive <NN>]
  [hw <HW> <address>]  [metric <NN>]  [mtu <NN>]
  [[-]trailers]  [[-]arp]  [[-]allmulti]
  [multicast]  [[-]promisc]
  [mem_start <NN>]  [io_addr <NN>]  [irq <NN>]  [media <type>]
  [txqueuelen <NN>]
  [[-]dynamic]
  [up|down] ...

  <HW>=Hardware Type.
  List of possible hardware types:
    loop (Local Loopback) slip (Serial Line IP) cslip (VJ Serial Line IP)
   slip6 (6-bit Serial Line IP) cslip6 (VJ 6-bit Serial Line IP) adaptive (Adaptive Serial
Line IP)
    strip (Metricom Starmode IP) ash (Ash) ether (Ethernet)
    tr (16/4 Mbps Token Ring) tr (16/4 Mbps Token Ring (New)) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) tunnel (IPIP Tunnel)
    ppp (Point-to-Point Protocol) hdlc ((Cisco)-HDLC) lapb (LAPB)
    arcnet (ARCnet) dlci (Frame Relay DLCI) frad (Frame Relay Access Device)
    sit (IPv6-in-IPv4) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) ec (Econet) x25 (generic X.25)
    infiniband (InfiniBand)
  <AF>=Address family. Default: inet
  List of possible address families:
    unix (UNIX Domain) inet (DARPA Internet) inet6 (IPv6)
    ax25 (AMPR AX.25) netrom (AMPR NET/ROM) rose (AMPR ROSE)
    ipx (Novell IPX) ddp (Appletalk DDP) ec (Econet)
    ash (Ash) x25 (CCITT X.25)
```

Aflarea adresei MAC din Windows

Adresa *MAC* a unui echipament poate fi determinată folosind comanda `IPCONFIG`, a cărei sintaxă completă este prezentată în caseta de pe pagina anterioară.

Un exemplu de utilizare pentru această comandă este prezentat tot într-o casetă.

Aflarea adresei MAC din Linux

În *Linux*, adresa *MAC* a unui echipament poate fi determinată folosind comanda `ifconfig`, a cărei sintaxă completă este prezentată în caseta de pe această pagină.

Cu ajutorul acestei comenzi putem obține[26] toate informațiile legate de interfețele active[27].

Un exemplu de utilizare este prezentat tot într-o casetă (pe pagina următoare).

Aflarea adresei MAC din Cisco IOS

Informațiilor legate de interfețe[28] sunt furnizate de comanda `SHOW INTERFACES`[29].

Un exemplu (doar parțial) este prezentat într-una dintre casete.

Modificabilitatea adresei MAC

Teoretic, adresa *MAC* este o valoare care nu poate fi modificată. Argumentele în favoarea nemodificabilității, sau, mai exact, în favoarea lipsei modificabilității facile sunt da-

[26] *în mod asemănător comenzii IPCONFIG de sub Windows*
[27] *inclusiv MAC-urile legate de interfețele de rețea*
[28] *în sensul celor afișate de IPCONFIG în Windows și ifconfig în Linux*
[29] *vezi [8], de unde este preluat exemplul*

Exemplu ifconfig

```
[root@server ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:C9:DC:88:A4
          inet addr:194.105.31.161  Bcast:194.105.31.255  Mask:255.255.255.0
          inet6 addr: fe80::2a0:c9ff:fedc:88a4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:190108802 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23596505 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:727058500 (693.3 MiB)  TX bytes:3711810236 (3.4 GiB)

eth1      Link encap:Ethernet  HWaddr 00:90:27:9F:CA:EA
          inet addr:86.104.31.78  Bcast:86.104.31.79  Mask:255.255.255.240
          inet6 addr: fe80::290:27ff:fe9f:caea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19925428 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27622255 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2325472658 (2.1 GiB)  TX bytes:3609928110 (3.3 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:510944 errors:0 dropped:0 overruns:0 frame:0
          TX packets:510944 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:550555697 (525.0 MiB)  TX bytes:550555697 (525.0 MiB)
```

te de scopul pentru care a fost definită această adresă: identificarea echipamentelor într-o rețea.

Totuși, această valoare poate fi modificată din mers[30] la unele echipamente, cum ar fi unele rutere[31], sau folosind anumite privilegii[32] pe sistemele *PC* clasice.

Trebuie remarcat faptul că, din punct de vedere teoretic, valoarea *MAC* a interfeței nu se modifică, ea fiind înscrisă în *controller*-ul interfeței de către fabricant. Ceea ce se modifică este acea valoare folosită de *driver*-ul interfeței / *kernel*-ul sistemului de operare cu ajutorul căreia sunt construite pachetele de rețea.

Modificarea adresei MAC în Windows

Modificarea *MAC*-ului în sisteme de tip *Windows XP*[33] se poate realiza numai dacă aveți drepturi de administrare pentru utilizatorul curent.

Pentru a modifica această valoare trebuie să procedăm astfel[34]:

- se accesează *Control Panel* → *System Properties* → *Network Adapter*;
- se efectuează un *click* pe placa de rețea în cauză[35];
- se accesează tabul *Advanced*;
- se caută opțiunea pentru specificarea adresei[36];
- în acest moment avem la dispoziție două opțiuni: *Not present*[37] și *Value*[38] (trebuie precizată o valoare explicită[39]).

Modificarea adresei MAC în Linux

Modificarea *MAC*-ului se realizează folosind instrucțiunea `ifconfif` cu parametrul `hw ether` *<adresă MAC>*.

Caseta care prezintă exemplul corespunzător ilustrează modificarea *MAC*-ului interfeței `eth1` de la `00:90:27:9F:CA:EA` la `BE:AC:AF:EA:00:00` imediat, fără a fi necesare *reboot*-ări sau reinițializări ale interfeței.

Afișarea tabelei MAC

Avem la dispoziție comenzi (atât în *Windows*, cât și în *Linux*) care permit vizualizarea conținutului tabelei *MAC*.

[30] *de exemplu, o pagină web de pe server-ul http implementat în echipament*
[31] *vezi [8], de unde este preluat exemplul*
[32] *root, administrator, după cum vom vedea ulterior*
[33] *Windows XP Home Edition, Windows XP Professional, Windows 2000, Windows 2000 Server, Windows 2003 Server etc.*
[34] *vezi captura de ecran de pe pagina următoare, cu ferestrele deschise în ordine*
[35] *în cazul nostru, o placă Intel® Pro/100*
[36] *în cazul nostru, Locally Administered Addess*
[37] *valoarea implicită a MAC-ului, înscrisă în controller-ul plăcii de rețea de către fabricant*
[38] *o valoare MAC nouă*
[39] *această valoare trebuie sa fie dată în hexazecimal, fără marcaje de separare (spațiu, două puncte, punct etc.)*

Exemplu show interfaces

```
Router# show interfaces

Ethernet 0 is up, line protocol is up
  Hardware is MCI Ethernet, address is 0000.0c00.750c (bia 0000.0c00.750c)
  Internet address is 131.108.28.8, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Last clearing of "show interface" counters 0:00:00
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 2000 bits/sec, 4 packets/sec
     1127576 packets input, 447251251 bytes, 0 no buffer
     Received 354125 broadcasts, 0 runts, 0 giants, 57186* throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5332142 packets output, 496316039 bytes, 0 underruns
     0 output errors, 432 collisions, 0 interface resets, 0 restarts
---More---
```

Afișarea tabelei MAC în Windows

Afișarea[40] tabelei *MAC* într-un sistem cu *Windows* se realizează folosind comanda `ARP` a cărei sintaxa este și ea prezentată într-o casetă.

Astfel, afișarea tabelei curente de *MAC*-uri se realizează așa cum se arată într-o altă casetă.

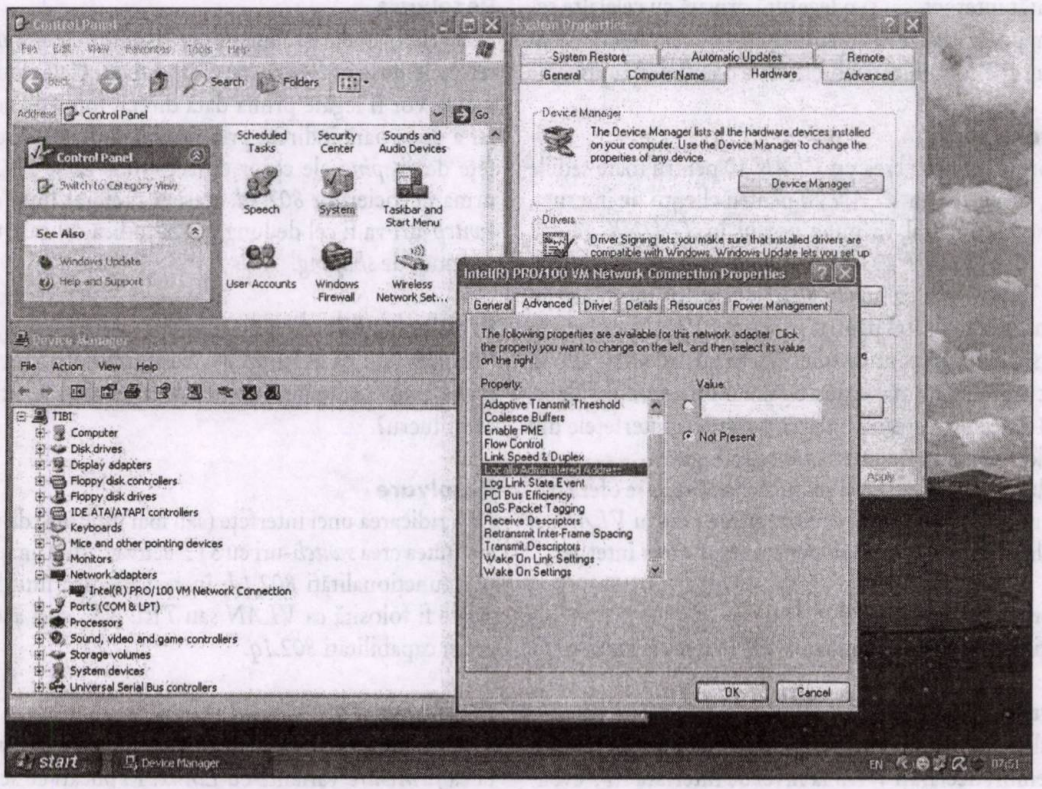
[40] *și manipularea, după cum vom vedea mai târziu (în partea a doua)*

Afișarea tabelei MAC în Linux

Afișarea[41] tabelei *MAC* într-un sistem cu *Linux* se realizează folosind comanda `arp` a cărei sintaxa este și ea într-o casetă.

Afișarea tabelei de *MAC*-uri curente se realizează conform exemplului dintr-o altă casetă.

[41] *și manipularea, după cum vom vedea mai târziu (în partea a doua)*



Modificarea adresei MAC în Linux

```
[root@server ~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:90:27:9F:CA:EA
          inet addr:86.104.31.78  Bcast:86.104.31.79  Mask:255.255.255.240
          inet6 addr: fe80::290:27ff:fe9f:caea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19927430 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27624411 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2325698805 (2.1 GiB)  TX bytes:3611156634 (3.3 GiB)

[root@server ~]# ifconfig eth1 hw ether BE:AC:AF:EA:00:00
[root@server ~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr BE:AC:AF:EA:00:00
          inet addr:86.104.31.78  Bcast:86.104.31.79  Mask:255.255.255.240
          inet6 addr: fe80::290:27ff:fe9f:caea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19927698 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27624748 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2325726927 (2.1 GiB)  TX bytes:3611189647 (3.3 GiB)
```

Rezolvările exercițiilor și problemelor propuse în episodul anterior

În cele ce urmează reamintim problemele propuse spre rezolvare în episodul anterior și prezentăm soluțiile acestora.

Problema #4

Furnizorul nostru de internet ne oferă la sediul central pe o fibră atât internet, cât și o legatură privată cu celelalte sedii. Cum poate realiza el acest lucru și ce avem noi de făcut pentru a rupe comunicația internet de comunicația privată?

Rezolvare

Furnizorul nostru va crea un *VLAN* 10 pentru toate sediile noastre, respectiv un *VLAN* 20 pentru clienții de internet.

La sediul central va pune portul în *TRUNK* pentru *VLAN* 10 și *VLAN* 20, oferindu-ne pe această legătură ambele *VLAN*-uri *tag*-ate cu *802.1q* care vor intra într-un echipament compatibil pentru selectarea *VLAN*-urilor sau într-un sistem *Linux*, unde interfața `eth0.20` va fi pusă în *bridge* cu `eth1` (am presupus că `eth0` este spre furnizor, iar `eth1` este spre rețeaua internă; aceasta și interfețele din sucursale trebuie să fie în aceeași rețea logică).

O altă variantă ar fi ca la sediul central să ni se ofere două legături de date pe fibră optică: prima pentru *VLAN*-ul cu sediile noastre, iar a doua pentru legatura la internet.

Problema #5

Explicați funcționarea sistemului *WDS* în rețelele radio *WiFi*.

Rezolvare

Sistemul *WDS* este, de fapt, un *bridge* care are *STP* activat între membri declarați *WDS* la nivel de interfață *RF*; eventual poate fi considerat și un *bridge* cu interfața *eth* a echipamentului.

Problema #6

Vă rugăm să vă imaginați un sistem de *shaping* transparent între două echipamente *switch* care au capabilități de *STP*.

Rezolvare

Între echipamente se introduce sistemul de *shaping* cu interfețele puse în *bridge*, cu *STP* activat. Cele două echipamente vor fi legate prima dată direct, folosind costul 100, iar a doua oară indirect prin intermediul celor două interfețe de *shaping* ale căror costuri sunt 10 și 10. Astfel, în urma negocierilor *802.1d*, traseul preferat între cele două *switch*-uri va fi cel de lungime 20, adică cel care trece prin sistemul de *shaping*.

Problema #7

Presupunând că aveți un *PC* cu *n* interfețe *Ethernet*, l-ați putea transforma într-un *switch* inteligent. Cum realizați acest lucru?

Rezolvare

Prin ridicarea unei interfețe (sau mai multora) de tip *bridge* am putea crea *switch*-uri cu *STP* activat, adică am putea adăuga funcționalități *802.1d*; în rest, fiecare interfață fizică poate fi folosită ca *VLAN* sau *TRUNK*, deci acest *switch* are și capabilități *802.1q*.

Problema #8

Aveți la dispoziție un echipament de tip *switch* care suportă ca *firmware* variante cu *Linux*. Explicați ce se întâmplă

Comanda ARP în Windows

```
C:\Documents and Settings\tibi>arp

Displays and modifies the IP-to-Physical address translation tables used by
address resolution protocol (ARP).

ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -a [inet_addr] [-N if_addr]

  -a            Displays current ARP entries by interrogating the current
                protocol data.  If inet_addr is specified, the IP and Physical
                addresses for only the specified computer are displayed.  If
                more than one network interface uses ARP, entries for each ARP
                table are displayed.
  -g            Same as -a.
  inet_addr     Specifies an internet address.
  -N if_addr    Displays the ARP entries for the network interface specified
                by if_addr.
  -d            Deletes the host specified by inet_addr. inet_addr may be
                wildcarded with * to delete all hosts.
  -s            Adds the host and associates the Internet address inet_addr
                with the Physical address eth_addr.  The Physical address is
                given as 6 hexadecimal bytes separated by hyphens. The entry
                is permanent.
  eth_addr      Specifies a physical address.
  if_addr       If present, this specifies the Internet address of the
                interface whose address translation table should be modified.
                If not present, the first applicable interface will be used.

Example:
  > arp -s 157.55.85.212   00-aa-00-62-c6-09  .... Adds a static entry.
  > arp -a                                    .... Displays the arp table.
```

când treceți un port în *VLAN*, respectiv când treceți un port în *TRUNK*.

Rezolvare

Dacă *switch*-ul suportă *Linux*, înseamnă că el se comportă ca un calculator cu *n* interfețe *eth*, deci este aplicabilă rezolvarea de la problema anterioară.

Problema #9

Aveți la dispoziție un sistem de *shaping* a traficului cam instabil (un *PC* cu probleme *hardware*) și un *switch* care știe *802.1d*. Cum proiectați un sistem de *shaping* cu o stabilitate mai mare folosind doar echipamentele pe care le aveți la dispoziție?

Rezolvare

Rezolvarea este expusă anterior în mod explicit în cazul în care avem două *switch*-uri.

Dacă avem un singur *switch*, atunci s-ar putea aplica rezolvarea anterioară, impunând condiția de a nu avea comutare între două porturi la nivelul procesorului echipamentului, ci doar via *802.1d*.

Exemplu arp în Windows

```
C:\Documents and Settings\tibi>arp -a
Interface: 86.104.31.77 --- 0x2
  Internet Address      Physical Address      Type
  86.104.31.78          be-ac-af-ea-00-00     dynamic
```

Comanda arp în Linux

```
[root@server ~]# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <hostname> [pub][nopub]    <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]              <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [temp][nopub] <-Add entry
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [netmask <nm>] pub  <-''-
  arp [-v]   [<HW>] [-i <if>] -Ds <hostname> <if> [netmask <nm>] pub      <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -e                      display (all) hosts in default (Linux) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           don't resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new entries from file or from /etc/ethers

  <HW>=Use '-H <hw>' to specify hardware address type. Default: ether
  List of possible hardware types (which support ARP):
    strip (Metricom Starmode IP) ash (Ash) ether (Ethernet)
    tr (16/4 Mbps Token Ring) tr (16/4 Mbps Token Ring (New)) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) arcnet (ARCnet)
    dlci (Frame Relay DLCI) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) x25 (generic X.25) infiniband (InfiniBand)
```

Va urma...

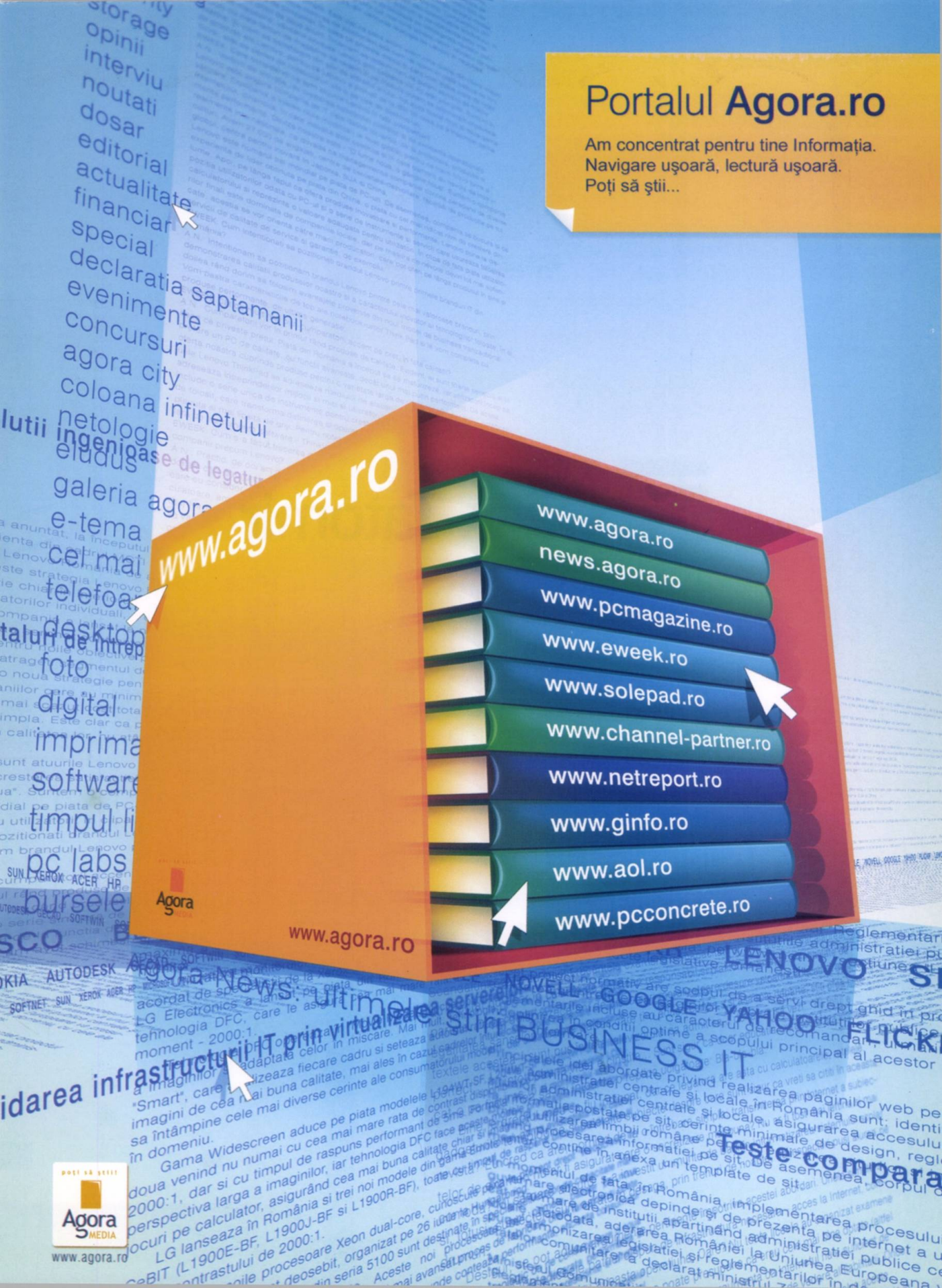
Datorită faptului că subiectul tratat este destul de amplu, am decis să împărțim episodul dedicat protocolului *ARP* în două părți. Cea de-a doua parte va fi publicată într-un număr viitor.

Bibliografie

1. Tiberiu Socaciu, *Informatica privită... altfel sau 16 chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft, 2006;
2. Tiberiu Socaciu, *VLAN, TRUNK și STP*, *GInfo* 17/1, ianuarie 2007, Editura Agora Media, Târgu Mureș;
3. Tiberiu Socaciu, *Tuneluri și bridge-uri*, *GInfo* 16/8, decembrie 2006, Editura Agora Media, Târgu Mureș;
4. Tiberiu Socaciu, *Limitări de viteză în rețele*, *GInfo* 16/6, octombrie 2006, Editura Agora Media, Târgu Mureș;
5. Tiberiu Socaciu, *Lecții de rețelistică*, vol. 1, în curs de apariție;
6. Tiberiu Socaciu, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura InfoData, 2006;
7. Tiberiu Socaciu, *O altfel de ... Informatică sau diverse chestiuni de matematică, backtracking, clustere de calculatoare, Linux, inteligență artificială și networking*, Editura InfoData, 2007;
8. Cisco, *documentație oficială pentru comanda* `show interfaces`, www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/inter_r/irshowin.htm#1017387.

Exemplu arp în Linux

```
[root@server ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
ro-cj01a-ubr1-ca-3-0-gw  ether   00:0E:D6:BD:C0:01   C                     eth0
86.104.31.76             ether   00:E0:50:02:2A:D9   C                     eth1
86.104.31.75             ether   00:E0:18:B1:B8:5B   C                     eth1
86.104.31.77             ether   00:30:05:29:60:9A   C                     eth
```


Portalul Agora.ro
Am concentrat pentru tine Informația.
Navigare ușoară, lectură ușoară.
Poți să știi...
www.agora.ro
www.agora.ro
news.agora.ro
www.pcmagazine.ro
www.eweek.ro
www.solepad.ro
www.channel-partner.ro
www.netreport.ro
www.ginfo.ro
www.aol.ro
www.pcconcrete.ro
www.agora.ro
Agora
www.agora.ro

Cabinetul
Călin Tatomir

Noi ştim cum

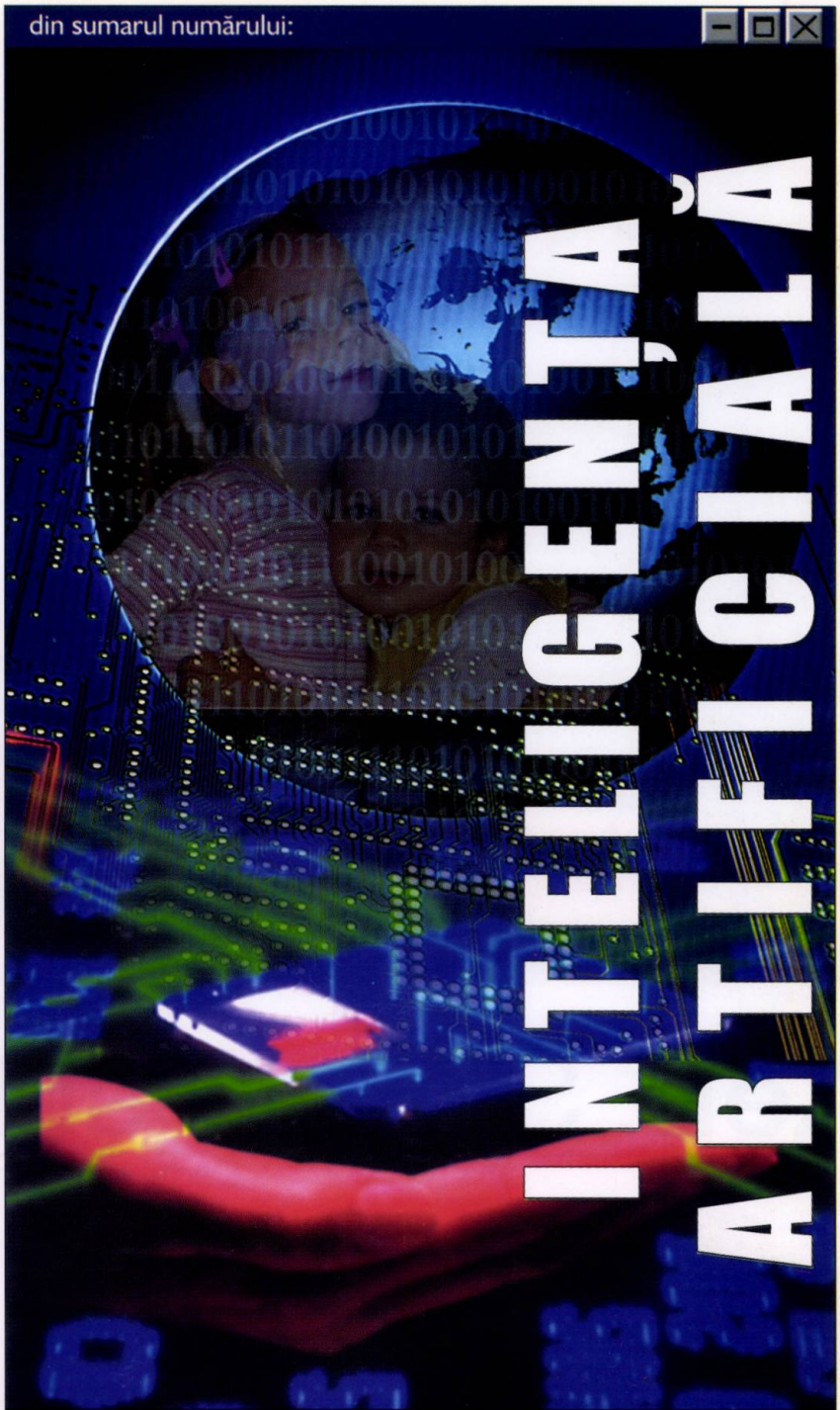
Martie 2007
Vol. 17/3
40.000 LEI / 4 LEI NOI

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

O publicație Agora Media

din sumarul numărului:



INTELIGENȚĂ ARTIFICIALĂ

probleme

Olimpiada LOCALĂ de informatică
Finala ACM 2007

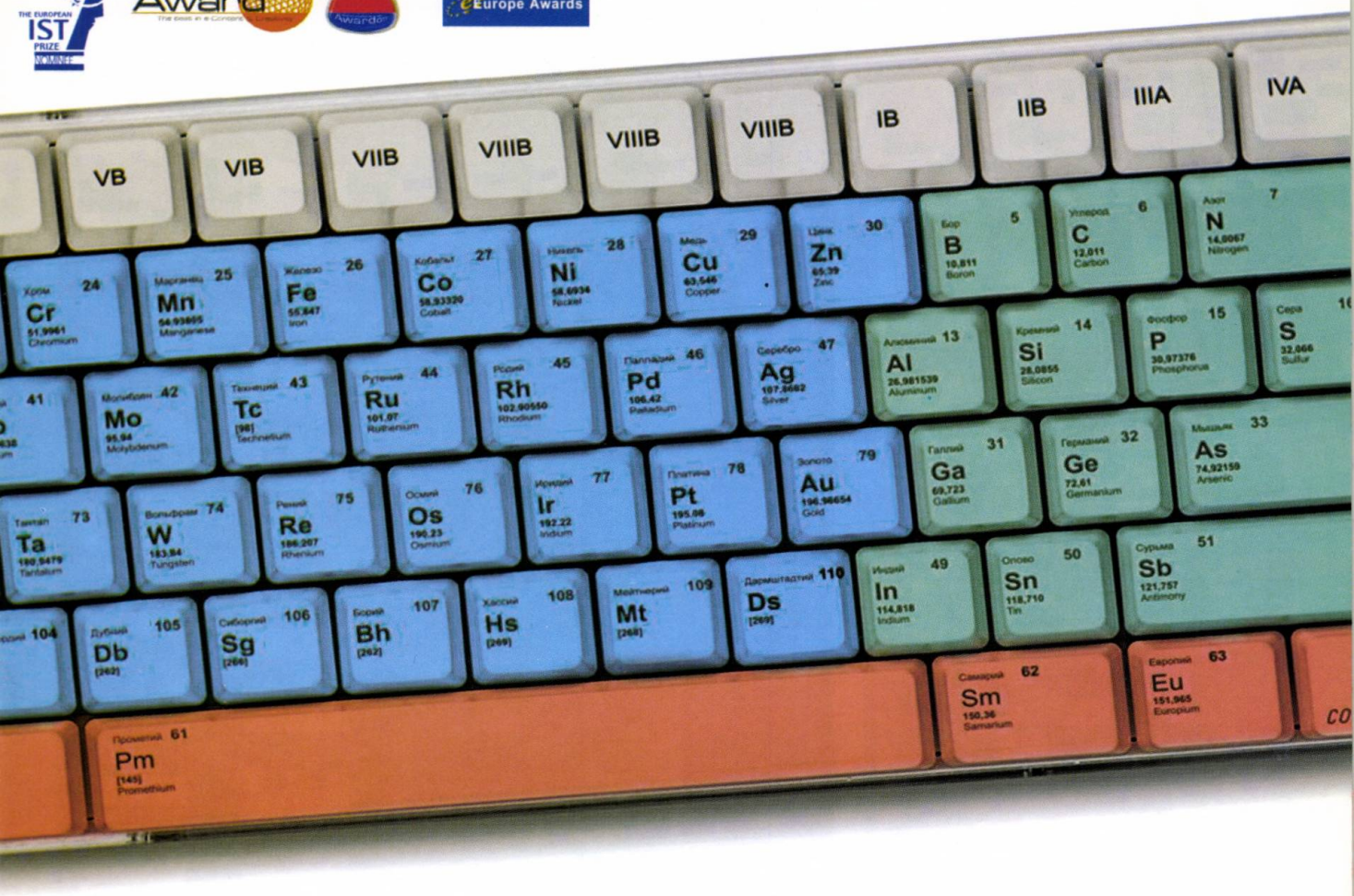
focus

Divizibilitate
Turnurile din Hanoi
Căutări în șiruri

serial

MAC și ARP (II)

5948373000645 03



AeL. Simplu şi la obiect.

Lecţii interactive pentru noua generaţie.

Orice întrebare are un răspuns!

Noutăţi, suport tehnic, competiţii,
forum : www.portal.edu.ro

Orice problemă are o soluţie!

Centrul de suport tehnic de luni până
vineri între orele 8.00-20.00
Telverde : 0800410444 (linie gratuită)

Orice cauţi poate fi găsit!

Ai nevoie de o anumită lecţie electronică?
Întâi de toate, caut-o în biblioteca virtuală
AeL. Odată găsită, descarc-o gratuit!

AeL
Built to teach intelli-gently

sei

SIVECO
ROMANIA SA

Vol. 17, Nr. 3 /2007

gazeta de informatică

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tiberiu Socaciu
Colaboratori:
conf. dr. *Iulian Pah*
prof. *Lucian Ilean*
Ionuț Chirilă
Alexandru Popa

Adresa pentru corespondență:
Computer Libris Agora,
Șerpuitoare 25, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/534.911, 0265/266.516
Fax: 0265/266.290

http://www.ginfo.ro
ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NET Report*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

O publicație Agora MEDIA

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu

Difuzare:
Laurențiu Nicolae
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

probleme.. PAG. 4

Olimpiada LOCALĂ de informatică. JUDEȚUL Cluj

Finala ACM 2007

focus.. PAG. 23

Aplicații ale proprietăților de DIVIZIBILITATE
Lucian ILEA, Iulian PAH

Problema GENERALIZATĂ a turnurilor din HANOI
Tiberiu SOCACIU

CĂUTĂRI în ȘIRURI
Alexandru POPA

e-learning.. PAG. 35

Construirea CUNOAȘTERII

tehno.. PAG. 36

PARCURGEREA unei structuri ARBORESCENTE de DIRECTOARE
Tiberiu SOCACIU

serial.. PAG. 40

MAC și ARP (II)
Tiberiu SOCACIU

INTELIGENȚA ARTIFICIALĂ între ADEVĂR și MIT
Ionuț CHIRILĂ

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

Singurele condiții pe care trebuie să le îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre și articolul să fie original (să nu mai fi fost publicat anterior și să nu fie o traducere a unui articol publicat de un alt autor).

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii.

Redacția GInfo nu își asumă nicio răspundere cu privire la informațiile prezentate în articole publicate.

Pentru articolele trimise la redacție, autorul cedează către GInfo dreptul exclusiv de publicare ale materialelor respective. Aceste articole nu pot fi publicate în alte publicații decât cu acordul scris al redacției sau în cazul în care redacția comunică autorului că articolul nu va fi publicat în revista GInfo.

Articolele publicate în revista GInfo nu pot fi preluate și publicate sub nicio formă decât cu acordul scris al redacției. Totuși, autorii articolelor pot publica integral sau parțial pe site-urile personale articolele cu GInfo, dar doar după 180 de zile de la publicare și cu menționarea faptului că articolul a fost publicat mai întâi în GInfo.

Pentru mai multe detalii vizitați:
http://www.ginfo.ro/revista/publicare.shtml

Cea mai bună introducere din lume în rețelele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine

EDIȚIA A PATRA

Rețele de calculatoare

ANDREW S. TANENBAUM

Cea mai bună introducere în arhitectura calculatoarelor - complet actualizată, reflectă ultimele schimbări din dinamica lume a calculatoarelor

Andrew S. Tanenbaum

ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR

Ediția a IV-a

SISTEME DE OPERARE MODERNE

EDIȚIA A DOUA

Andrew S. Tanenbaum

Această carte de referință, bazată pe exemple concrete, oferă informații practice de primă mână pentru înțelegerea și construirea sistemelor de operare moderne.

THOMAS H. CORMEN CHARLES E. LEISERSON RONALD R. RIVEST

INTRODUCERE ÎN ALGORITMI

Cartea tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, fiind binevenita elevilor, studenților și profesorilor

CĂRȚILE ESENȚIALE ▸

...cărțile care îți însoțesc cariera

Comandă-le pe toate și le iei cu 20% mai ieftin.
Iar dacă ești student, cu 30%.

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

Concursurile din 2007

Sezonul de competiții din acest an a început. Faza locală și faza județeană au avut deja loc, națională se apropie cu pași repezi, s-a terminat și cu finala ACM și mai avem BOI, CEOI și IOI...

Prima competiție importantă a anului a fost finala *ACM* de la *Tokyo*. Au câștigat polonezii...

Interesantă a fost departajarea pentru poziția a treia. Două echipe aveau același număr de probleme rezolvate și același număr de puncte de penalizare. Departajarea s-a efectuat pe baza momentului trimiterii ultimei soluții corecte. Au fost mai rapizi rușii...

Din nefericire, românii care au reprezentat *Universitatea din București* au încheiat competiția pe la mijlocul plutonului.

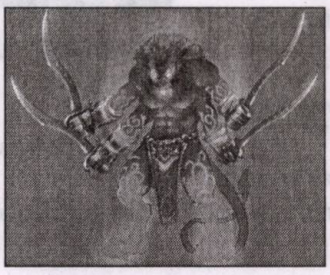
Despre faza locală a olimpiadei de informatică nu sunt multe de spus. Subiectele au fost unice la nivelul fiecărui județ și au fost aleși cei care vor participa la faza județeană.

Faza județeană a avut și ea loc pe 10 martie cu subiecte unice la nivel național.

Fericiții concurenți care vor reprezentanta județele țării se vor întâlni la *Cluj-Napoca* în prima jumătate a lunii aprilie. Atunci vor fi aleși și membrii lotului național care vor participa la barajele de selecție pentru alegerea echipelor care vor reprezenta *România* la concursurile internaționale.

Despre olimpiadele internaționale din acest an știm cu siguranță că *Olimpiada Internațională de Informatică* se va desfășura în capitala *Croației*, *Zadar*, în a doua jumătate a lunii august.

De fapt, despre *IOI* se știu deja și locațiile următoarelor trei ediții. În 2008 cei mai buni informaticieni se vor întâlni la *Alexandria*, în *Egipt*. În 2009 *Olimpiada Internațională de Informatică* va reveni în *Bulgaria* (țara care a găzduit prima ediție), la *Plovdiv*. *Bulgaria* va deveni atunci singura țară care a găzduit două ediții ale concursului. În 2010 participanții vor vizita orașul canadian *Waterloo* din statul *Ontario*.

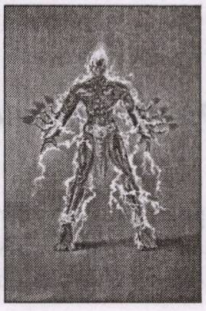


Despre *Olimpiada de Informatică a Europei Centrale* știm doar că ar trebui să se desfășoare la *Brno*, în *Cehia*. Astfel acest oraș ar deveni al doilea, după *Cluj-Napoca*, care găzduiește două ediții ale *CEOI*-ului. Exista și varianta ca *România* să găzduiască ediția din acest an a competiției.

Cât despre *Balcaniadă*, ca de obicei, nu se știe nimic sigur...

Vă dorim mult succes la concursurile la care veți mai participa în acest an!

Mai menționez faptul că în acest număr nivelurile de dificultate ale articolelor sunt ilustrate de vrăjitori și creaturile magice ale acestora (ordinea este gremlin, gargui, golem, magician, gin, rakshasa și titan - imaginile sunt prezentate și pe această pagină, gremlinul nu apare decât pe această pagină, neexistând articole cu niel de dificultate foarte scăzut) și că în numărul următor vom trece la *dwarfi*.



Mihai Scorțaru

OLI 2007, Cluj

Olimpiada LOCALĂ de informatică. Județul CLUJ

Faza locală a olimpiadei de informatică s-a desfășurat, în județul Cluj, pe data de 16 februarie pentru elevii de gimnaziu și pe data de 17 februarie pentru cei de liceu; subiectele au fost unice în întregul județ. Vă prezentăm în continuare enunțurile tuturor celor 14 probleme propuse spre rezolvare, puse la dispoziție de către coordonatorul concursului, dl. prof. Mugurel Butyka de la Colegiul Național "George Barițiu" din Cluj-Napoca.

Clasa a V-a

P070301: Numere impare

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Se citește de la tastatură un număr natural n, cuprins între 10 și 30000. Să se afișeze pe ecran cele mai apropiate două numere impare de numărul dat.

Exemple

Intrare
20

Ieșire
19 21

Intrare
33

Ieșire
31 35

Explicații

- 19 este cel mai mare număr impar mai mic decât 20;
- 21 este cel mai mic număr impar mai mare decât 20;
- 31 este cel mai mare număr impar mai mic decât 33;
- 35 este cel mai mic număr impar mai mare decât 33.

P070302: Invidie

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Spunem că un număr natural nenul a ***invidiază*** un număr natural nenul b dacă suma cifrelor lui a este ***strict mai mică*** decât suma cifrelor lui b.

Să se scrie un program care citește două numere naturale nenule a și b, având cel mult patru cifre, și afișează DA dacă numărul a invidiază numărul b sau NU în caz contrar.

Exemple

Intrare
123
273

Ieșire
DA

Intrare
2456
3365

Ieșire
NU

Explicații

- suma cifrelor numărului 123 este 6;
- suma cifrelor numărului 723 este 12;
- suma cifrelor numărului 2456 este 17;
- suma cifrelor numărului 3365 este 17.

Clasa a VI-a

P070303: Vocale

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Se citește de la tastatură un șir format din cel mult 100 de caractere.

Să se afișeze numărul de vocale și numărul de consoane întâlnite.

Exemple

Intrare

```
Acesta este un exemplu.
```

Ieșire

```
10
9
```

Intrare

```
Olimpiada Locala de Informatica
```

Ieșire

```
14
14
```

P070304: Șir de numere

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Se citește de la tastatură un șir format din n ($9 < n < 100$) numere întregi care conțin cel mult patru cifre.

Să se determine și să se afișeze pe ecran numărul de numere strict pozitive, numărul de numere strict negative, numărul de numere pare și numărul de numere impare din șirul citit.

Exemple

Intrare

```
15
17 0 34 -15 673 723 -53 -54 -63 10 -74 -9999
                                    -53 0 0
```

Ieșire

```
5
7
7
8
```

Intrare

```
10
-23 234 0 0 -198 1 56 790 0 1
```

Ieșire

```
5
2
7
3
```

Explicații

Pentru primul exemplu:

- cele cinci numere strict pozitive sunt 17, 34, 673, 723 și 10;
- cele șapte numere strict negative sunt -15, -53, -54, -63, -74, -9999 și -53;
- cele șapte numere pare sunt 0, 34, -54, 10, -74, 0 și 0;
- cele opt numere impare sunt 17, -15, 673, 723, -53, -63, -9999 și -53.

Pentru al doilea exemplu:

- cele cinci numere strict pozitive sunt 234, 1, 56, 790 și 1;
- cele două numere strict negative sunt -23 și -198;
- cele șapte numere pare sunt 234, 0, 0, -198, 56, 790 și 0;
- cele trei numere impare sunt -23, 1 și 1.

Notă

Elementele șirului sunt date pe o singură linie; în primul exemplu prezentat în revistă, acestea apar pe două linii din motive de spațiu.

Clasa a VII-a

P070305: Note la test

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Dintre elevii unei clase, n ($5 < n < 30$) elevi au participat la un test la informatică. Notele obținute de ei sunt numere naturale cuprinse între 1 și 10.

Se citesc de la tastatură numărul n și apoi numele și notele celor n elevi ai clasei.

Să se afișeze nota maximă din clasă și elevii care au obținut această notă.

Exemple

Intrare

```
7
Ana 8
Vasile 7
Maria 5
Gigel 4
Elena 8
Mihaela 9
Costel 6
```

Ieșire

```
9
Mihaela
```

Intrare

```
8
Ana 8
Vasile 7
Maria 5
Gigel 4
Elena 7
Mihaela 8
Costel 6
Nelu 8
```

Ieșire

```
8
Ana
Mihaela
Costel
```

P070306: Mărgele

Adrian Miclea și Mirela Pop
Colegiul Național "Emil Racoviță" Cluj-Napoca

Adriana are un șirag format din n mărgele, (n este un număr natural mai mare decât 2 și cel mult egal cu 1000000000).

Mărgelele sunt înșirate pe o sfoară și sunt de două culori, unele albe, altele roșii.

Ea dorește să afle câte mărgele se află între prima mărgică de culoare roșie și ultima mărgică din șir care este tot roșie.

Dacă ar începe să le numere durează prea mult și poate încurca numărătoarea și atunci ea vă roagă pe voi să scrieți un program care citește de la tastatură numărul n, o succesiune de n litere (litera `a` codifică o mărgică albă și litera `r` codifică o mărgică roșie) care codifică culorile mărgelelor și afișează pe ecran numărul de mărgele cuprinse între prima și ultima mărgică de culoare roșie (se numără și aceste două mărgele).

Exemple

Intrare
```
10
aarrrraara
```

Ieșire
```
7
```

Intrare
```
8
aaaaaaaa
```

Ieșire
```
0
```

Intrare
```
8
aaraaaaa
```

Ieșire
```
0
```

Explicații

- pentru primul exemplu, prima mărgică roșie se află pe poziția a treia a șirului, iar ultima mărgică roșie se află pe poziția a noua a șirului;
- pentru al doilea exemplu, nu avem nicio mărgică roșie;
- pentru al treilea exemplu, avem o singură mărgică roșie.

Clasa a VIII-a

P070307: Numere șmechere

Adrian Pintea
Colegiul Național "Andrei Mureșanu" Dej

Spunem că numărul natural nenul și neprim a este mai *șmecher* decât numărul natural nenul și neprim b dacă mulțimea divizorilor proprii ai numărului b este strict inclusă în mulțimea divizorilor proprii ai numărului a.

Să se scrie un program care citește un vector x format din n ($1 < n < 50$) numere naturale nenule și neprime, un vector y format din m ($1 < m < 50$) numere naturale nenule și neprime, și afișează pentru fiecare număr din vectorul y numărul de numere din vectorul x față de care este mai șmecher.

Exemple

Intrare
```
3
12 81 54
4
6 54 20 10
```

Ieșire
```
0 1 0 0
```

Intrare
```
4
4 66 35 21
3
24 70 105
```
Ieșire
```
1 2 2
```

Explicații

Pentru primul exemplu avem:

- mulțimea divizorilor proprii ai numărului 12 este $dp(12) = \{2, 3, 4, 6\}$;
- mulțimea divizorilor proprii ai numărului 81 este $dp(81) = \{3, 9, 27\}$;
- mulțimea divizorilor proprii ai numărului 54 este $dp(54) = \{2, 3, 6, 9, 18, 27\}$;
- mulțimea divizorilor proprii ai numărului 6 este $dp(6) = \{2, 3\}$;
- mulțimea divizorilor proprii ai numărului 20 este $dp(20) = \{2, 4, 5, 10\}$;
- mulțimea divizorilor proprii ai numărului 10 este $dp(10) = \{2, 5\}$;
- numărul 6 nu este mai șmecher decât niciun număr din vectorul x;
- numărul 54 este mai șmecher decât numărul 81;
- numărul 20 nu este mai șmecher decât niciun număr din vectorul x;
- numărul 10 nu este mai șmecher decât niciun număr din vectorul x.

Pentru al doilea exemplu avem:

- mulțimea divizorilor proprii ai numărului 4 este $dp(4) = \{2\}$;
- mulțimea divizorilor proprii ai numărului 66 este $dp(66) = \{2, 3, 6, 11, 22, 33\}$;
- mulțimea divizorilor proprii ai numărului 35 este $dp(35) = \{5, 7\}$;

- mulțimea divizorilor proprii ai numărului 21 este $dp(21) = \{3, 7\}$;
- mulțimea divizorilor proprii ai numărului 24 este $dp(24) = \{2, 3, 4, 6, 8, 12\}$;
- mulțimea divizorilor proprii ai numărului 70 este $dp(70) = \{2, 5, 7, 10, 14, 35\}$;
- mulțimea divizorilor proprii ai numărului 105 este $dp(105) = \{3, 5, 7, 15, 21, 35\}$;
- numărul 24 este mai șmecher decât numărul 4;
- numărul 70 este mai șmecher decât numărul 4 și decât numărul 35;
- numărul 105 este mai șmecher decât numărul 35 și decât numărul 21.

P070308: Supraveghere

Adrian Pintea
Colegiul Național "Andrei Mureșanu" Dej

O firmă de pază are de supravegheat o zonă marcată cu borne kilometrice.

Prima bornă are numărul 1, iar următoarele sunt numerotate cu 2, 3, etc., ultima fiind 100.

Firma dispune de n paznici, ($2 < n \leq 100$, număr natural) și pentru fiecare paznic se cunosc cele două borne kilometrice între care va face supravegherea.

Firma dorește să construiască o clădire în zona pe care o au comună toți cei n paznici.

Scrieți un program care citește de la tastatură numărul n al paznicilor, apoi n perechi de forma b_1 b_2, unde b_1 și b_2 sunt bornele kilometrice pentru fiecare paznic, și afișează bornele kilometrice între care se poate realiza construcția, astfel încât aceasta să fie supravegheată de toți cei n paznici ai firmei.

Dacă nu există o astfel de zonă, se va afișa valoarea 0.

Exemple

Intrare
```
5
2 100
4 90
80 95
85 95
4 98
```

Ieșire
```
85 90
```

Intrare
```
4
1 25
20 40
28 35
1 55
```

Ieșire
```
0
```

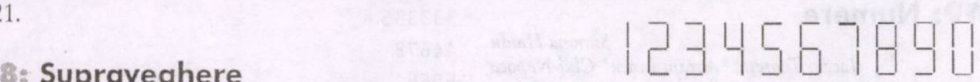
Clasa a IX-a

P070309: Pedeapsa

Tudor Mihoc
Liceul de Informatică "Tiberiu Popoviciu" Cluj-Napoca

Alex este elev în clasa a V-a și învață aritmetică. Șmecher, a vrut să rezolve exercițiile la ora de matematică cu un calculator de buzunar.

Profesorul l-a prins; vrând să îl pedepsească l-a atenționat în primul rând asupra modului cum sunt afișate cifrele pe ecranul calculatorului de buzunar folosind liniuțe:



Așadar, cifra 1 ar fi reprezentată folosind două liniuțe, cifra 2 folosind cinci liniuțe etc.).

Profesorul l-a întrebat apoi pe *Alex* care este cel mai mic număr ce poate fi reprezentat pe ecran folosind exact n liniuțe (niciuna în plus sau în minus)? Dar cel mai mare număr de acest tip? Văzând că băiatul se descurcă, profesorul i-a cerut să calculeze și suma primelor m numere naturale cu această proprietate.

Scrieți un program care să ușureze munca băiatului!

Date de intrare

Fișierul de intrare **PEDEAPSA.IN** conține pe prima linie numerele naturale n și m, despărțite de un spațiu.

Date de ieșire

Fișierul de ieșire **PEDEAPSA.OUT** va conține pe prima linie numărul minim cu proprietatea cerută, pe a doua linie numărul maxim cu proprietatea cerută și pe a treia linie suma cerută.

Restricții și precizări

- $1 < n \leq 20$;
- $1 \leq m \leq 2000$;
- dacă există mai puțin de m numere cu proprietatea cerută, se vor însuma toate aceste numere, indiferent de numărul lor.

Exemplu

PEDEAPSA.IN
```
5 4
```

PEDEAPSA.OUT
```
2
71
27
```

Explicație

- cel mai mic număr care poate fi reprezentat folosind cinci liniuțe este 2;
- cel mai mare număr care poate fi reprezentat folosind cinci liniuțe este 71;

• suma căutată este 2 + 3 + 5 + 17 = 27 (2, 3, 5 și 17 sunt primele patru numere care pot fi reprezentate folosind cinci liniuțe)

Notă

Pentru determinarea corectă a valorii minime se acordă 20% din punctaj, pentru determinarea corectă a valorii maxime se acordă 30% din punctaj, iar pentru determinarea corectă a sumei se acordă 50% din punctaj.

Timp de execuție: o secundă/test

P070310: Numere

Simona Haidu
Liceul Teoretic "Avram Iancu" Cluj-Napoca

Vlăduț este elev în clasa I. El știe că există numere cu multe cifre, dar de-abia a învățat cifrele și operația de adunare cu 1, astfel încât rezultatul să fie mai mic decât 10.

Doamna învățătoare le dă copiilor din clasă o fișă care conține mai multe numere naturale formate din cel mult nouă cifre fiecare și le propune următorul exercițiu: din numerele de pe fișă să formeze zece grupe de numere. Prima grupă va conține toate numerele în care cifra 0 apare de un număr maxim de ori, a doua grupă va conține toate numerele în care cifra 1 apare de un număr maxim de ori și așa mai departe, până la a zecea grupă, care va conține toate numerele în care cifra 9 apare de un număr maxim de ori.

Acasă, *Vlăduț* se ambiționează să refacă exercițiul pornind de la mult mai multe numere, dar operația devine din ce în ce mai dificilă. Realizați un program care să-l ajute pe *Vlăduț* să se verifice, chiar dacă pornește de la un număr mare de numere!

Date de intrare

Fișierul de intrare `NUMERE.IN` conține pe prima linie numărul n, reprezentând numărul de numere de pe fișa de lucru și pe următoarele n linii câte un număr natural format din cel mult nouă cifre, reprezentând numerele de pe fișă.

Date de ieșire

Fișierul de ieșire `NUMERE.OUT` va conține soluția problemei formată din cele zece grupe de numere; fiecare grupă se va afișa pe o linie separată (pe prima linie grupa numerelor cu număr maxim de cifre 0, pe a doua linie grupa numerelor cu număr maxim de cifre de 1 etc.). În cadrul unei grupe, numerele sunt separate prin spațiu.

Dacă o grupă nu conține niciun număr (de exemplu, anumite cifre nu apar deloc în niciunul dintre numerele de pe fișă), atunci linia corespunzătoare din fișier va fi vidă.

Restricții și precizări

• $1 < n \leq 15000$, număr natural;
• cele n numere de pe fișă sunt numere naturale formate din cel mult nouă cifre;
• în fișierul de ieșire un număr poate apărea în cadrul mai multor grupe;
• în cadrul unei grupe, ordinea de apariție a numerelor în grupă este dată de ordinea de apariție a numerelor în fișierul de intrare.

Exemplu

`NUMERE.IN`

```
10
11123
2234
1156
1117
2223
333335
44678
5566
55789
6666
```

`NUMERE.OUT`

```
11123 1117
2223
333335
44678
5566 55789
6666
1117 44678 55789
44678 55789
55789
```

Explicație

• prima linie din fișierul de ieșire este vidă deoarece nu există numere care să conțină cifra 0;
• pe a doua linie apar numerele în care cifra 1 apare de un număr maxim de ori (de trei ori), în ordinea dată de apariția în fișierul de intrare, separate printr-un spațiu;
• pe a treia linie apare singurul număr care conține numărul maxim de cifre 2;
...

Timp de execuție: o secundă/test

Clasa a X-a

P070311: Dreptunghiuri

Mihaela Giurgea
Liceul de Informatică "Tiberiu Popoviciu" Cluj-Napoca

Gigel învață grafica pe calculator. Pentru început a învățat să deseneze dreptunghiuri de diferite mărimi, cu laturile paralele cu axele de coordonate. După ce desenează n astfel de dreptunghiuri, *Gigel* încearcă să le grupeze.

Oare câte dintre dreptunghiuri nu se intersectează cu niciun alt dreptunghi? Câte se intersectează cu un singur dreptunghi? Câte se intersectează cu două dreptunghiuri? ... Câte se intersectează cu n - 1 dreptunghiuri? (toate celelalte)

Ajutați-l pe *Gigel* să răspundă la aceste întrebări!

Date de intrare

Fișierul de intrare **DREPT.IN** conține pe prima linie un număr natural n, iar pe următoarele n linii câte patru numere naturale reprezentând coordonatele colțurilor stânga-sus (abscisă, ordonată) și respectiv dreapta-jos (abscisă, ordonată) pentru fiecare dreptunghi, separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **DREPT.OUT** va conține n numere naturale. Primul număr va reprezenta numărul dreptunghiurilor care nu se intersectează cu niciun alt dreptunghi, al doilea număr va reprezenta numărul dreptunghiurilor care se intersectează cu un singur dreptunghi și așa mai departe până la ultimul număr (al n - lea) care va reprezenta numărul dreptunghiurilor care se intersectează cu toate celelalte dreptunghiuri.

Numerele se vor scrie pe prima linie, separate prin câte un spațiu.

Restricții și precizări

- $1 < n \leq 1000$, număr natural;
- ecranul pe care sunt desenate dreptunghiurile are colțul din stânga-sus în punctul de coordonate (0, 0) și colțul din dreapta-jos în punctul de coordonate (1000, 1000);
- considerăm că *două dreptunghiuri se intersectează dacă suprafețele lor au cel puțin un punct comun* (chenarul unui dreptunghi este considerat a face parte din suprafața acestuia).

Exemplu

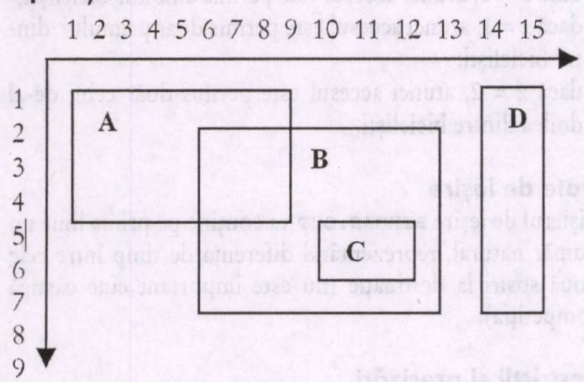
DREPT.IN

```
4
1 1 9 5
6 2 13 8
10 5 12 7
14 1 15 8
```

DREPT.OUT

```
1 2 1 0
```

Explicație



- există un dreptunghi care nu se intersectează cu niciun alt dreptunghi (D).
- există două dreptunghiuri care se intersectează cu un alt dreptunghi (A, C).
- există un dreptunghi care se intersectează cu două alte dreptunghiuri (B).
- nu există niciun dreptunghiuri care se intersectează cu toate celelalte trei dreptunghiuri.

Notă

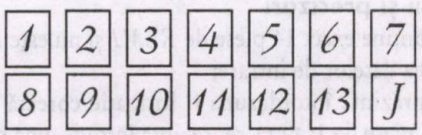
Pentru determinarea corectă a valorii minime se acordă 20% din punctaj, pentru determinarea corectă a valorii maxime se acordă 30% din punctaj, iar pentru determinarea corectă a sumei se acordă 50% din punctaj.

Timp de execuție: o secundă/test

P070312: Remi

Dana Vaida
Liceul Teoretic "Avram Iancu" Cluj-Napoca

Jocul de *REMI* folosește piese pe care sunt înscrise toate numerele de la 1 la 13, în următoarele patru culori: roșu, negru, galben și albastru. În plus, mai există exact două piese înscrise cu *J*, numite *Joly-Joker* care au calitatea de a înlocui orice altă piesă. Piesele înscrise cu negru, respectiv *Joly-Joker* sunt următoarele:



Pe tabla de joc există exact 14 piese de *REMI*, dispuse în diferite formații. Culoarea pieselor nu este precizată, dar se presupune că întotdeauna este cea corectă. Formațiile care se punctează în joc (numite în continuare formații corecte) sunt de următoarele două tipuri:

a. formațiile constituite din cel puțin trei numere consecutive, în ordine crescătoare; de exemplu: `123` (formată din piesele 1, 2 și 3), `891011` (formată din piesele 8, 9, 10 și 11) sau `7J9` (formată din piesele 7, *Joly-Joker* - care înlocuiește piesa 8 - și 9); este de asemenea permisă formația de cel puțin trei piese în ordine crescătoare care conține o piesă 1 după piesa 13, de exemplu `12131` (formată din piesele 12, 13 și 1)

b. formațiile constituite din trei sau patru numere identice (deoarece există maxim patru culori); de exemplu: `1111` (patru piese de 1), `555` (trei piese de 5), `111111` (trei piese de 11), `13131313` (patru piese de 13) sau `77JJ` (două piese de 7 în două culori și două piese *Joly-Joker* care înlocuiesc două piese de 7 de celelalte două culori)

Punctajul formațiilor corecte se calculează știind că:

- piesele 2, 3, 4, 5, 6, 7, 8, și 9 au valoarea pe care o arată;
- piesele 10, 11, 12 și 13 au valoarea 10;
- piesa 1 are valoarea 1 dacă se află înaintea lui 2 și valoarea 10 dacă se află lângă alte cifre de 1 sau după 13; de exemplu formația `1234` are valoarea 1 + 2 + 3 + 4 = 10, forma-

ția 1111 are valoarea 4 · 10 = 40, iar formația 9101112131 are valoarea 9 + 5 · 10 = 59;
- *Joly-Joker* are valoarea piesei pe care o înlocuiește.

Date de intrare

Fișierul de intrare **REMI.IN** conține o singură linie pe care se află un șir de caractere reprezentând formații de pe tabla de *REMI*. Formațiile din șir sunt despărțite printr-un spațiu și ele pot fi corecte (în cazul în care corespund descrierii de la punctele *a.* sau *b.*) sau incorecte (în caz contrar). În cadrul unei formații (corecte sau nu) nu există spații sau alte caractere cu excepția celor permise în joc (*J* și numerele cuprinse între 1 și 13).

Date de ieșire

Fișierul de ieșire **REMI.OUT** va conține o singură linie pe care se vor tipări trei numere separate între ele prin câte un spațiu.

Primul număr reprezintă numărul de formații corecte descrise la punctul *a.*, al doilea număr reprezintă numărul de formații corecte descrise la punctul *b.*, iar al treilea număr reprezintă punctajul total al formațiilor corecte de pe tablă.

Restricții și precizări

- tabla conține exact 14 piese de *REMI* și nu este necesară validarea datelor de intrare;
- 111 reprezintă întotdeauna o formație corectă formată din trei piese de 1; 1111 reprezintă întotdeauna o formație corectă formată din patru piese de 1; 111111 reprezintă întotdeauna o formație corectă formată din trei piese de 11; 11111111 reprezintă întotdeauna o formație corectă formată din patru piese de 11; 11111 și 1111111 sunt formații incorecte;
- datele de intrare nu conțin formații care pot fi interpretate ambiguu (de exemplu J10J, care poate fi interpretată ca fiind 91011 sau 101010); dacă o formație care conține *Joly-Joker* poate reprezenta o formație corectă, atunci întotdeauna această formație reprezintă una corectă.

Exemple

REMI.IN
```
23 444 5667 56789
```

REMI.OUT
```
1 1 47
```

REMI.IN
```
67889 121212 9J11 12131
```

REMI.OUT
```
2 1 89
```

Explicații

- pentru primul exemplu există o singură formație corectă de tipul *a.*, 56789, cu valoarea 5 + 6 + 7 + 8 + 9 = 35 și o singură formație corectă de tipul *b.*, 444, cu valoarea 3 · 4 = 12, deci valoarea totală este 35 + 12 = 47;
- pentru al doilea exemplu există două formații corecte de tipul *a.*, 9J11, cu valoarea 9 + 10 + 10 = 29, respectiv 12131, cu valoarea 3 · 10 = 30 și o singură formație corectă de tipul *b.*, 121212, cu valoarea 3 · 10 = 30, deci valoarea totală este 29 + 30 + 30 = 89.

Timp de execuție: o secundă/test

Clasele a XI-a și a XII-a

P070313: Bicicliști

Lucian Ilea
Colegiul Național "Emil Racoviță" Cluj-Napoca

Doi amici, *A* și *B*, bicicliști pasionați, se iau la întrecere pe străzile unui oraș. Intersecțiile orașului sunt numerotate de la 1 la *n* (*n* este număr natural).

Cele *m* străzi care conectează câte două intersecții permit deplasarea în ambele sensuri. Bicicliștii se deplasează cu viteze egale.

Bicicliștii pornesc în același timp din intersecția 1 și trebuie să ajungă cât mai repede în intersecția *n*. Deși deplasarea pe unele străzi poate fi interzisă pentru unul sau altul dintre cei doi, fiecare din ei poate să ajungă la destinație.

Se cere să se determine diferența de timp la sosire între cei doi bicicliști, presupunând că fiecare dintre ei alege cel mai bun traseu disponibil pentru el.

Date de intrare

Fișierul de intrare **BIKERS.IN** conține pe prima linie două numere naturale *n* și *m*, separate printr-un spațiu, reprezentând numărul intersecțiilor, respectiv numărul străzilor din oraș.

Pe următoarele *m* linii se află câte patru numere naturale *x*, *y*, *t* și *z*, separate prin câte un spațiu. Numerele *x* și *y* desemnează intersecții între care există drum, *t* este timpul necesar deplasării între intersecțiile *x* și *y*, iar *z* este un număr natural care codifică restricțiile de deplasare pe strada respectivă, astfel:
- dacă *z* = 0, atunci accesul este permis ambilor bicicliști;
- dacă *z* = 1, atunci accesul este permis doar primului dintre bicicliști;
- dacă *z* = 2, atunci accesul este permis doar celui de-al doilea dintre bicicliști.

Date de ieșire

Fișierul de ieșire **BIKERS.OUT** va conține pe prima linie un număr natural, reprezentând diferența de timp între cele două sosiri la destinație (nu este important cine câștigă competiția).

Restricții și precizări

- $2 \le n \le 50$;
- $1 \le t \le 100$;

- în setul de date nu există străzi care să fie descrise de mai multe ori și între oricare două intersecții există cel mult o stradă;
- datele de intrare sunt corecte (problema are întotdeauna soluție).

Exemplu

BIKERS.IN

```
6 9
1 2 1 0
1 3 10 2
2 3 1 2
3 4 1 0
4 5 2 0
5 6 3 2
2 4 4 1
2 5 1 1
4 6 10 1
```

BIKERS.OUT

```
6
```

Explicație

- biciclistul *A* poate ajunge cel mai repede din intersecția 1 în intersecția 6 urmând pe rând șirul de intersecții 1, 2, 5, 4 și 6, într-un timp de 1 + 1 + 2 + 10 = 14.
- biciclistul *B* poate ajunge cel mai repede din intersecția 1 în intersecția 6 urmând pe rând șirul de intersecții 1, 2, 3, 4, 5 și 6, într-un timp de 1 + 1 + 1 + 2 + 3 = 8.
- diferența între timpii de sosire (număr natural!) este 6.

Timp de execuție: o secundă/test

P070314: Falanga macedoneană

Mugurel Butyka
Colegiul Național "George Barițiu" Cluj-Napoca

Alexandru Macedon își conduce trupele în lupta împotriva împăratului persan *Darius*. Trupele sale sunt aliniate pe câmpul de bătălie în două rânduri, astfel încât o singură poziție din aceste rânduri este liberă.

Alexandru realizează că pentru a obține o victorie decisivă este necesar să-și permute trupele, dar trebuie să realizeze aceasta cât mai rapid și inteligent.

Astfel, pentru a nu slăbi puterea de luptă a formației, permutarea trupelor din configurația inițială în cea finală se va face printr-o succesiune de mutări. O mutare constă în deplasarea unei trupe în poziția liberă.

De asemenea, *Alexandru* știe că trupele din primul rând sunt mai solicitate și dorește ca acestea să efectueze un număr minim de mutări.

Ajutați-l pe *Alexandru* să-și deplaseze trupele din configurația inițială în cea finală în număr minim de mutări și solicitând cât mai puțin trupele din primul rând (dintre mai multe soluții care necesită un număr minim de mutări pentru a ajunge de la configurația inițială la cea finală se alege una care conține un număr minim de mutări ale trupelor din primul rând).

Date de intrare

Fișierul de intrare **FALANGA.IN** conține pe prima linie un număr natural *n*, reprezentând numărul trupelor disponibile.

Pe următoarele două linii este codificată configurația inițială a trupelor, astfel: trupele sunt identificate prin numere de la 1 la *n*, iar poziția liberă prin valoarea 0. Fiecare linie conține (*n* + 1) / 2 valori separate prin câte un spațiu.

Pe următoarele două linii este codificată configurația finală a trupelor în mod similar configurației inițiale.

Date de ieșire

Fișierul de ieșire **FALANGA.OUT** va conține o singură soluție a problemei codificată așa cum se arată în cele ce urmează.

Pe prima linie se află un număr natural reprezentând numărul minim de mutări în care se poate ajunge la configurația finală pornind de la configurația inițială, respectând cerințele din enunț.

Pe a doua linie se află un șir de numere naturale separate prin câte un spațiu care conține de la stânga la dreapta numărul de ordine al trupelor în ordinea în care trebuie deplasate.

Restricții și precizări

- $1 \le n < 100$, număr natural impar;
- configurația inițială diferă întotdeauna de cea finală;
- datele de intrare sunt corecte (problema are întotdeauna soluție).

Exemplu

FALANGA.IN

```
7
7 3 0 2
1 4 6 5
3 2 6 5
4 0 1 7
```

FALANGA.OUT

```
8
5 7 3 2 5 6 1 4
```

Explicație

În configurația inițială este liberă a treia poziție din primul rând, iar în cea finală a doua poziție din al doilea rând.

Ordinea mutării trupelor este:

- trupa 5 urcă din al doilea rând pe poziția liberă
- trupa 7 coboară în poziția eliberată de trupa 5
- trupa 3 se mută la stânga în poziția eliberată de trupa 7

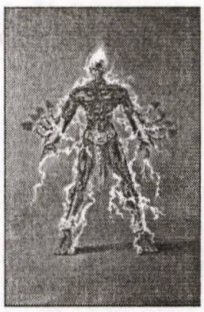
...

Varianta `6 1 4 7 3 2 5 7` necesită tot opt mutări, dar trupele din primul rând efectuează în total patru mutări față de trei în soluția corectă.

O altă variantă optimă ar fi `6 1 4 5 7 3 2 5`.

Timp de execuție: o secundă/test

Tokyo, Japonia



Finala ACM 2007

Vă prezentăm în continuare enunțurile celor zece probleme cu care s-au confruntat reprezentanții celor 88 de universități care au participat la faza finală din acest an a concursului de programare ACM.

P070315: Calcule cosangvine

Sângele fiecărei persoane are două caracteristici numite alele *ABO*. Fiecare dintre aceste caracteristici este reprezentată prin una dintre literele `A`, `B` sau `O`.

Astfel, putem avea șase combinații posibile ale acestor alele, fiecare dintre ele corespunzând unei grupe sangvine, așa cum se arată în tabelul următor.

Combinație	Grupă sangvină
AA	A
AB	AB
AO	A
BB	B
BO	B
OO	O

În mod asemănător, fiecare persoană are două alele pentru *Rh*-ul sângelui, reprezentate prin caracterele + și -. Pentru o persoană care are *Rh* pozitiv (Rh+) cel puțin una dintre alele este + (pot fi ambele). Persoanele cu *Rh* negativ au ambele alele -.

Tipul de sânge al unei persoane este o combinație între grupa sangvină și *Rh*. Tipul de sânge este notat prin adăugarea sufixului + sau - la grupa sangvină. Câteva exemple ar putea fi `A+`, `AB-` sau `O-`.

Tipurile de sânge se moștenesc: fiecare părinte biologic "donează" una dintre alelele sale *ABO* (aleasă aleator dintre cele două) și una dintre alele *Rh*.

Așadar, două alele *ABO* și două alele *Rh* ale părinților determină tipul de sânge al copilului. De exemplu, dacă ambii părinți ai unui copil au tipul de sânge `A-`, atunci tipul de sânge al copilului va fi fie `A-`, fie `O-`. Un copil ai cărui părinți au tipurile de sânge `A+` și `B+` poate avea oricare dintre tipurile de sânge.

Pentru această problemă, veți avea la dispoziție tipul de sânge al ambilor părinți sau tipul de sânge al unui părinte și al copilului și va trebui să determinați tipurile de sânge posibile ale copilului (în primul caz) sau ale celuilalt părinte (în al doilea caz). Este posibil ca mulțimea determinată să fie vidă.

Notă

Pentru această problemă grupa sangvină *O* va fi reprezentată prin litera mare `O` și nu prin cifra `0` (zero).

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **`blood.in`**.

Fiecare set de date va consta dintr-o linie pe care se vor afla trei șiruri de caractere reprezentând tipul de sânge al primului părinte, tipul de sânge al celui de-al doilea părinte și tipul de sânge al copilului. Una dintre cele trei valori va lipsi, fiind înlocuită cu un semn de întrebare (`?`).

Pentru a îmbunătăți lizibilitatea, pot apărea spații între șirurile care reprezintă tipurile de sânge, dar nu și în interiorul acestor șiruri.

Ultimul set de date va fi urmat de o linie pe care se vor afla literele `E`, `N` și `D` separate prin spații.

Date de ieșire

Pentru fiecare set de date, la ieșirea standard se va scrie numărul de ordine al setului de date (numerotarea începe de la 1) și tipurile de sânge ale celor doi părinți și cel al copilului.

Dacă nu există niciun tip de sânge posibil pentru un părinte, în poziția corespunzătoare va fi scris mesajul `IMPOSSIBLE`.

Dacă există mai multe variante posibile pentru tipul de sânge al unui părinte sau al copilului, lista acestora va fi inclusă între acolade. Ordinea în care apar variantele în interiorul acoladelor nu este importantă.

În exemplul de mai jos apar mai multe variante posibile ale ieșirii. Va trebui ca programele să genereze ieșiri asemănătoare.

Exemplu

`blood.in`

```
O+ O- ?
O+ ? OAB-
AB+ ?
AB+ ? O+
E N D
```

Ieșire

```
Case 1: O+ O- {O+, O-}
Case 2: O+ {A-, A+, B-, B+, O-, O+} O
Case 3: AB- AB+ {A+, A-, B+, B-, AB+, AB-}
Case 4: AB+ IMPOSSIBLE O+
```

P070316: Containere

Un terminal dintr-un port maritim înmagazinează mari containere care vor fi încărcate pe diferite vase pentru a fi transportate peste graniță.

Containerele care sosesc pe străzi sau pe calea ferată intră în terminal pe măsură ce sosesc, în ordine și sunt așezate unele peste altele, în mai multe stive.

Vasele transportă un număr mare de containere. Timpul necesar încărcării unui vas depinde, parțial, de locațiile containerelor care trebuie încărcate. Timpul de încărcare crește dacă anumite containere nu se află în partea de sus a unei stive și pot fi încărcate numai după ce containerele aflate deasupra lor sunt înlăturate.

Este nevoie de un plan pentru terminal astfel încât timpul de încărcare să scadă. Planul trebuie să permită fiecărui vas să se încarce preluând doar containere din vârfurile stivelor.

De asemenea, trebuie minimizat numărul total al stivelor de containere.

Pentru această problemă cunoaștem ordinea în care trebuie încărcate vasele și ordinea în care sosesc containerele. Fiecare vas este reprezentat printr-o literă mare (de la A la Z, inclusiv) și navele vor fi încărcate în ordine alfabetică.

Fiecare container este etichetat cu o literă mare care reprezintă vasul pe care trebuie încârcat acesta.

Nu există nicio limită pentru numărul de containere dintr-o stivă.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **containers.in**.

Fiecare set de date constă într-o singură linie care conține între una și 1000 de litere mari care reprezintă containerele în ordinea în care sosesc.

De exemplu, linia ABAC indică faptul că vor sosi patru cotainere care trebuie încărcate în vasele A, B, A, respectiv C. După ce sosesc toate containerele acestea trebuie încărcate în vase în ordine strict crescătoare: mai întâi este încărcat vasul A, apoi vasul B și apoi vasul C.

Ultimul set de date va fi urmat de o linie pe care se va afla doar cuvântul end.

Date de ieșite

Pentru fiecare set de date, programul dumneavoastră va trebui să scrie la ieșirea standard, pe câte o linie, numărul de ordine al setului de date (numerotarea începe de la 1) și numărul minim de stive necesare pentru păstrarea containerelor înainte de încărcare.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

containers.in

```
A
CBACBACBACBACBA
CCCCBBBBAAAA
ACMICPC
end
```

Ieșire

```
Case 1: 1
Case 2: 3
Case 3: 1
Case 4: 4
```

P070317: Mare premiu

Sunteți proiectantul șef al unei curse stradale care se va desfășura într-o regiune deluroasă. Traseul constă în N segmente de dreaptă legate între ele; cu alte cuvinte, punctul final al celui de-al k - lea segment este punctul inițial al celui de-al $(k + 1)$ - lea segment.

În timpul planificări, aceste segmente sunt plasate pe o suprafață plană. Așadar, extremitățile lor sunt identificate prin coordonate carteziene bidimensionale. Punctul inițial al primului segment este dat de originea sistemului de coordonate.

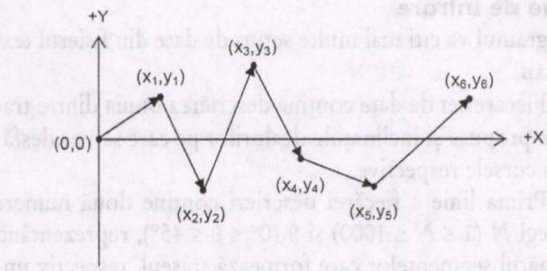


Figura 1

În figura 1 este prezentată o imagine plană a unui circuit format din șase segmente. Cursa va avea loc pe un deal. Pentru simplitate, vom considera că unghiul dintre suprafața dealului și planul orizontal este θ, așa cum se arată în figura 2.

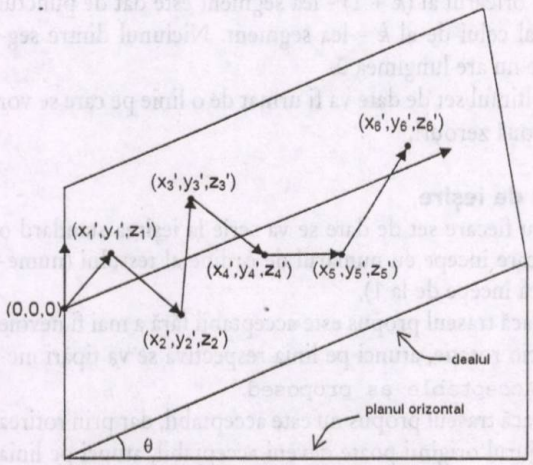


Figura 2

Planul dealului și planul orizontal se intersectează de-a lungul axei y. Fiecărei perechi de coordonate carteziene bidimensionale (x_i, y_i) îi corespunde o tripletă de coordonate carteziene tridimensionale(x'_i, y'_i, z'_i), unde z'_i reprezintă înălțimea punctului final al celui de-al i - lea segment de dreaptă. Înălțimea punctului de origine este 0.

Această cursă este destinată începătorilor, deci traseul nu poate include segmente care să coboare dealul. Așadar, dacă înălțimea punctului final al celui de-al k - lea segment este z'_k, atunci înălțimea oricărui segment care urmează după cel de-al k - lea nu poate fi mai mică decât z'_k. Cu alte cuvinte, putem spune că avem $z'_k \le z'_m$ pentru $m \ge k$.

Dacă un traseu propus include segmente care coboară dealul, ar putea fi posibilă transformarea traseului într-unul care nu conține astfel de segmente prin rotirea reprezentării planare în jurul originii, fără a modifica unghiurile dintre segmentele consecutive.

Totuși, ar putea exista trasee propuse care nu pot deveni acceptabile în urma unei astfel de rotații.

Pentru această problemă va trebui să determinați dacă anumite trasee propuse sunt acceptabile (nu conțin niciun segment care coboară dealul). Dacă traseul nu este acceptabil, atunci trebuie determinat cel mai mic unghi cu care trebui rotit acesta pentru a deveni acceptabil (dacă este posibil).

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `gp.in`.

Fiecare set de date conține descrierea unuia dintre traseele propuse și înclinațiile dealurilor pe care se vor desfășura cursele respective.

Prima linie a fiecărei descrieri conține două numere întregi N ($1 \le N \le 1000$) și θ ($0° \le \theta \le 45°$), reprezentând numărul segmentelor care formează traseul, respectiv unghiul, exprimat în grade, dintre deal și planul orizontal.

Fiecare dintre următoarele N linii conține câte o pereche de numere întregi (x_i, y_i), $(1 \le i \le N)$, care reprezintă coordonatele punctelor finale ale segmentelor de dreaptă care descriu traseul. Punctul inițial al primului segment este dat de originea sistemului de coordonate și punctul inițial al oricărui al $(k + 1)$ - lea segment este dat de punctul final al celui de-al k - lea segment. Niciunul dintre segmente nu are lungimea 0.

Ultimul set de date va fi urmat de o linie pe care se vor afla două zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1).

Dacă traseul propus este acceptabil fără a mai fi nevoie de nicio rotație, atunci pe linia respectivă se va tipări mesajul `Acceptable as proposed`.

Dacă traseul propus nu este acceptabil, dar prin rotirea sa în jurul originii poate deveni acceptabil, atunci pe linia respectivă se va tipări mesajul `Acceptable after clockwise rotation of X degrees` sau mesajul `Acceptable after counterclockwise rotation of X degrees`. Valoarea `X` trebuie să fie un număr fără semn.

Pentru această problemă vom considera că o rotație în sensul acelor de ceasornic (engl. *clockwise*) înseamnă rotirea axei y pozitive înspre axa x pozitivă.

Dacă traseul poate deveni acceptabil atât printr-o rotire în sensul acelo de ceasornic, cât și print-o rotire în sens invers acelor de ceasornic, atunci va fi aleasă cea pentru care unghiul este mai mic.

Dacă atât pentru o rotație în sensul acelor de ceasornic, cât și pentru una în sens invers acelor de ceasornic, unghiul de rotire este același, poate fi aleasă oricare dintre cele două rotații.

Dacă nu există posibilitatea rotirii traseului astfel încât acesta să devină acceptabil, mesajul tipărit va fi `Unacceptable`.

Valorile unghiurilor vor fi tipărite folosind două zecimale exacte.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

gp.in

```
2 45
10 10
0 1
2 0
1 1
2 0
3 45
10 10
1 10
5 6
0 0
```

Ieșire

```
Case 1: Acceptable after clockwise rotation of
                                    131.99 degrees

Case 2: Acceptable as proposed

Case 3: Unacceptable
```

Notă

În cadrul datelor de ieșire din exemplul de mai sus, mesajul corespunzător primului set de date trebuie să apară pe o singură linie. În revistă acesta apare pe două linii din motive de spațiu.

P070318: Circuite Jacquard

Excentricul sculptor *Albrecht Caravaggio Mondrian* s-a inspirat din istoria calculatoarelor pentru a crea opere de artă pe care le-a numit *circuite Jacquard*.

Fiecare dintre lucrările sale constă din mai multe plăci de circuite integrate (de formă poligonală), fiecare având aceeași formă, dar la scale diferirte, care sunt legate între ele prin sârme sau fire, obținându-se o structură tridimensională.

În figura 3 este prezentat un exemplu care constă în două niveluri de formă pentagonală.

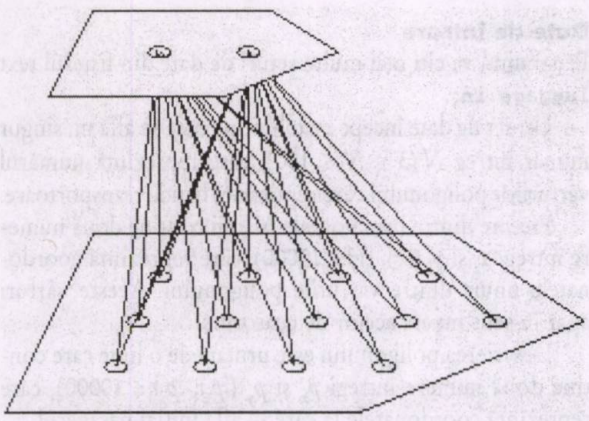


Figura 3

A.C.M. (așa cum îi spun prietenii) și-a bazat operele pe cartelele perforate folosite în războaiele de țesut *Jacquard* (care au fost adaptate mai târziu de către *Charles Babbage* pentru motorul său analitic) și pe aspectul grilelor folosite de obicei în interfațarea circuitelor.

Plăcile de circuite integrate folosite în sculpturi au forma unor poligoane (aceste poligoane sunt laticiale). Un poligon laticial este definit ca fiind un ciclu format din segmente de dreaptă care nu se autointersectează, care unesc puncte ale căror coordonate sunt numere întregi și printre care oricare două segmente consecutive nu sunt paralele.

Pentru orice poligon liticial P, există un poligon laticial mai mic care are aceeași formă și aceeași orientare ca și P; acest poligon este denumit *origine*.

Poligoanele mai mici care au aceeași formă și aceeași orientare cu poligonul P sunt numite predecesoare ale poligomului P, iar poligoanele mai mari care au aceeași formă și aceeași orientare cu poligonul P sunt numite succesoare ale poligonului P (vezi figura 4).

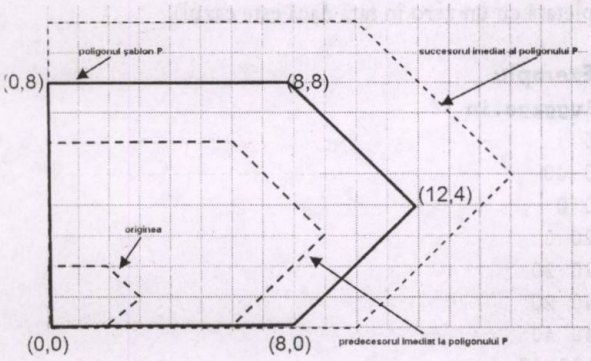


Figura 4

Pentru a crea una dintre sculpturile sale, *A.C.M.* începe prin a alege la întâmplare N puncte laticiale, după care desenează un șablon unind punctele alese.

Trebuie observat faptul că nu este neapărat necesar ca toate cele N puncte să fie vârfurile poligonului laticial. De exemplu, este posibil ca trei sau mai multe puncte consecutive să fie coliniare.

Fie P poligonul laticial determinat de șablon. Sculptorul determină originea corespunzătoare poligonului P, alege numărul M al nivelurilor pe care dorește să le construiască și creează primele M plăci de circuite integrate poligonale din categoria aleasă (originea și următoarele M - 1 poligoane succesoare).

Fiecare punct laticial aflat strict în interiorul oricăruia dintre aceste poligoane laticiale conține un orificiu în care se for întâlni firele sau sârmele.

Partea dificilă a procesului constă în legarea tuturor firelor și a tuturor sârmelor care se întâlnesc într-un anumit orificiu.

Mai mult, câteva dintre faimoasele sculpturi miniaturizate ale lui *A.C.M.* au fost construite folosind tehnici care implică procese nanotehnologice și conțin sute de mii de niveluri.

Mondrian și-ar dori să determine, pe baza unei forme pologonale și a numărului de niveluri, câte orificii vor exista în sculptura rezultată.

Va trebui să scrieți un program pentru a-l ajuta. De exemplu, scultura din figura 3 conține 15 orificii.

Vom considera că toate orificiile de pe plăcile care formează sculptura au diametrul 0.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `jacquard.in`.

Fiecare set de date va începe cu o linie pe care se vor afla două numere întregi pozitive N ($3 \le N \le 1000$) și M ($1 \le M \le 1000000$). N reprezintă numărul punctelor laticiale alese de *Mondrian*, iar M reprezintă numărul nivelurilor sculpturii pe care o va crea.

Fiecare dintre următoarele N linii conține două numere întregi x și y ($|x|, |y| \le 1000000$), care reprezintă coordonatele unuia dintre punctele alese de *Mondrian*. Punctele sunt date fie în sensul acelor de ceasornic, fie în sens invers acelor de ceasornic.

Ultimul set de date va fi urmat de o linie pe care se vor afla două zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de numărul total al orificiilor unei sculpturi cu M niveluri care pornește de la originea poligonului dat.

Această valoare nu va depăși în nicio situație cea mai mare valoare posibilă a unui număr întreg cu semn reprezentat pe 64 de biți.

Exemplu

jacquard.in

```
5 2
0 0
8 0
12 4
8 8
0 8
3 2
-1 -1
3 1
5 -1
0 0
```

Ieșire

```
Case 1: 15
Case 2: 2
```

P070319: Recuperarea bagajelor

Recuperarea bagajelor după un zbor cu avionul poate fi departe de o chestiune banală. Geamantanele și sacoșele își fac apariția pe o bandă transportoare și sute de pasageri luptă pentru un loc avantajos din care să își poată recupera mai ușor bagajele.

Recent, oficialii aeroportului *Narita* au decis să eficientizeze acest proces. Înaintea reproiectării zonelor în care pasagerii își recuperează bagajele, ei au nevoie de un program de simulare care să determine cum se comportă un pasager obișnuit în momentul în care își recuperează bagajele.

Simularea se bazează pe faptul că pasagerii vor alege întotdeauna un drum format din segmente de linii drepte pentru a ajunge la bagajele lor cât mai repede posibil.

Pentru această problemă, o bandă transportoare este modelată sub forma unui poligon simplu. Un bagaj va apărea într-un anumit punct al benzii și apoi se va mișca odată cu banda, având o viteză constantă.

Un pasager este poziționat inițial într-un punct aflat în afara poligonului care reprezintă banda transportoare. Imediat după apariția bagajului său, pasagerul începe să se miște cu o viteză constantă (mai mare decât cea cu care se deplasează bagajul) pentru a-și recupera bagajul.

Traseul pasagerului, care nu se poate intersecta cu banda transportoare, dar o poate atinge, îl va duce pe acesta, cât mai repede posibil, în aceeași poziție cu bagajul său.

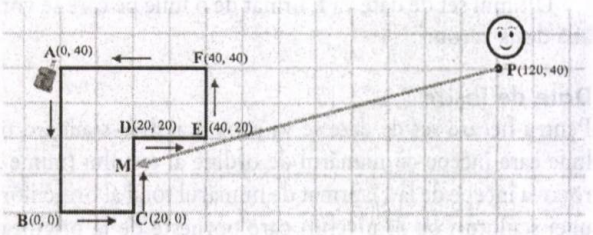


Figura 5

În figura 5 banda transportoare este reprezentată de poligonul *ABCDEF*. Bagajul apare în colțul din stânga-sus (punctul *A*) și se mișcă în sens invers acelor de ceasornic în jurul poligonului (direcția este cea indicată de săgeți).

Pasagerul pornește din punctul *P* și se deplasează pe traseul care îl va duce cât mai repede în același loc cu bagajul său (punctul *M* din figură). Traseul este ilustrat de săgeata deschisă la culoare. Imaginea din figură corespunde primului set de date din exemplul de mai jos.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **luggage.in**.

Un set de date începe cu o linie pe care se află un singur număr întreg N ($3 \le N \le 100$), care reprezintă numărul vârfurilor poligonului care reprezintă banda transportoare.

Fiecare dintre următoarele N linii conține două numere întregi x_i și y_i ($|x_i|, |y_i| \le 10000$), care reprezintă coordonatele unuia dintre vârfurile poligonului. Aceste vârfuri apar în sens invers acelor de ceasornic.

Descrierea poligonului este urmată de o linie care conține două numere întregi p_x și p_y ($|p_x|, |p_y| \le 10000$), care reprezintă coordonatele la care se află inițial pasagerul.

Ultima linie corespunzătoare unui set de date conține două numere întregi V_L și V_P ($0 < V_L < V_P \le 10000$), care reprezintă viteza cu care se deplasează bagajul, respectiv viteza cu care se deplasează pasagerul.

Toate coordonatele sunt exprimate în metri și toate vitezele sunt exprimate în metri pe minut.

Puteți presupune că poziția inițială a pasagerului este în afara poligonului care reprezintă banda transportoare și că bagajul se va deplasa pe banda trabsportoare în sens invers acelor de ceasornic, pornind de la primul vârf al poligonului.

După descrierea fiecărui set de date, urmează câte o linie vidă.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de timpul minim în care pasagerul poate ajunge la bagajul său.

Va trebui utilizat formatul din exemplul de mai jos (cu minute și secunde separate prin caracterul ':') și timpul va fi rotunjit la cea mai apropiată secundă. Valoarea corespunzătoare secundelor va avea întotdeauna două cifre (completată cu un zero în față dacă este cazul).

Exemplu

luggage.in

```
6
0 40
0 0
20 0
20 20
40 20
40 40
120 40
70 100
```

```
4
0 0
10 0
10 10
0 10
100 100
10 11
0
```

Ieșire

```
Case 1: Time = 1:02
Case 2: Time = 12:36
```

P070320: Joc cu bile

Pentru acest joc avem M bile pe o tablă împărțită în $N \times N$ pătrățele ale căror laturi sunt egale cu unitatea și M dintre acestea conțin găuri. Bilele și găurile sunt numerotate de la 1 la M. Scopul jocului este de a introduce fiecare bilă în gaura care are același număr.

Tabla poate conține mai multe ziduri. Fiecare zid are lungimea egală cu unitatea și este amplasat între două pătrățele adiacente.

Două pătrățele sunt considerate a fi adiacente dacă și numai dacă au o latură comună.

La începutul jocului toate bilele se află pe tablă, fiecare într-un pătrățel separat. O mutare constă în ridicarea tablei într-o anumită direcție. Ca urmare a ridicării toate bilele se deplasează înspre latura opusă celei ridicate, ele alunecând până când ating un zid, cad într-o gaură sau ating o altă bilă.

Bilele respectă următoarele restricții:

- bilele nu pot sări peste ziduri, peste alte bile sau peste găurile libere;
- bilele nu pot părăsi tabla (există un zid pe margine care înconjoară întreaga tablă);
- un pătrățel poate conține o singură bilă la un moment dat;
- când o bilă ajunge într-un pătrățel în care se află o gaură, bila acde în gaura respectivă; în acest moment gaura este acoperită și alte bile se pot rostogoli pe deasupra ei.

Jocul se încheie în momentul în care toate bilele au căzut în găurile corespunzătoare.

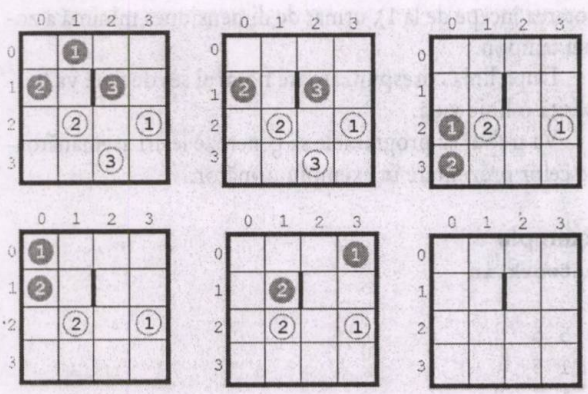


Figura 6

În figura 6 este prezentată soluția unui joc pentru o tablă de dimensiuni 4×4, care conține trei bile, trei găuri și un zid. Soluția constă în cinci mutări: ridicarea laturii din dreapta (est), ridicarea laturii de sus (nord), ridicarea laturii de jos (sud), ridicarea laturii din stânga (vest) și ridicarea laturii de sus (nord).

Programul dumneavoastră va trebui să determine numărul minim de mutări prin care toate bilele ajung în găurile corespunzătoare, dacă o astfel de secvență există.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `marble.in`.

Un set de date începe cu o linie pe care se află trei numere întregi: dimensiunea N ($2 \le N \le 4$) a tablei, numărul M ($M > 0$) al bilelor și numărul W al zidurilor.

Fiecare dintre următoarele $2 \cdot M$ linii conține două numere întregi; primul indică o linie, iar cel de-al doilea o coloană. Primele M linii reprezintă pozițiile bilelor; primei bile îi corespunde prima dintre aceste linii, celei de-a doua linii îi corespunde a doua bilă și așa mai departe. Ultimele M linii reprezintă pozițiile găurilor; primei găuri îi corespunde prima dintre aceste linii, celei de-a doua găuri îi corespunde a doua bilă și așa mai departe.

În sfârșit, următoarele W linii descriu pozițiile zidurilor. Fiecare dintre aceste linii conține patru numere întregi; prima pereche de numere indică linia și coloana pătrățelului aflat de o parte a zidului, iar cea de-a doua pereche indică linia și coloana pătrățelului aflat de cealaltă parte a zidului. Liniile și coloanele sunt numerotate de la 0 la $N - 1$.

Ultimul set de date este urmat de o linie pe care se află trei zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de numărul minim de mutări necesare câștigării jocului.

În cazul în care jocul nu poate fi câștigat, în locul numărului de mutări va fi scris mesajul `impossible`.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

`marble.in`

```
4 3 1
0 1
1 0
1 2
2 3
2 1
3 2
1 1 1 2
3 2 2
```

```
0 0
0 1
0 2
2 0
2 0 1 0
2 0 2 1
0 0 0
```

Ieșire

```
Case 1: 5 moves

Case 2: impossible
```

P070321: Rețea

O rețea cu rutare a pachetelor manipulează informațiile în unități mici, împărțind mesajele mai lungi în mai multe pachete.

Deși fiecare pachet traversează rețeaua pe trasee diferite și pachetele pot ajunge la destinație la momente de timp diferite și nu în ordinea inițială, calculatorul care trebuie să primească mesajul reconstruiește corect mesajul inițial.

Calculatorul căruia îi este destinat mesajul folosește o memorie tampon pentru a păstra pachetele care nu sosesc în ordinea corectă.

Va trebui să scrieți un program care va determina dimensiunea minimă a zonei tampon care este necesară pentru a reconstrui mesajele care sosesc, cunoscând numărul N al mesajelor, numărul M al pachetelor care formează fiecare mesaj, dimensiunea fiecărui mesaj și ordinea în care sosesc pachetele.

În momentul în care sosește un pachet, acesta este fie transferat în zona tampon, fie este transferat direct la ieșire. Toate pachetele din zona tampon pot fi transferate la ieșire în orice moment.

Se spune că un pachet *trece de zona tampon* în momentul în care este transferat la ieșire și că un mesaj *trece de zona tampon* în momentul în care toate pachetele care îl formează au trecut de zona tampon.

Pachetele oricărui mesaj trebuie să fie ordonate astfel încât datele din secvența de pachete care trec de zona tampon să fie în ordinea corectă.

De exemplu, pachetul care conține octeții numerotați de la 3 la 5 ai unui mesaj va trebui să treacă de zona tampon înaintea pachetului care conține octeții numerotați de la 6 la 10 ai aceluiași mesaj.

Mesajele pot trece de zona tampon în orice ordine, dar toate pachetele care formează un anumit mesaj trebuie să treacă de zona tampon consecutiv și în ordinea corectă (dar nu neapărat la același moment de timp).

Trebuie observat faptul că, spre deosebire de sistemele reale, pentru această problemă putem să ne uităm înainte la toate pachetele care vor veni și putem lua decizii în consecință.

Pachetele constau dintr-un antet și din datele propriu-zise. Antetul conține trei numere: numărul de ordine al mesajului, poziția în mesaj a primului octet din pachet și poziția în mesaj a ultimului octet din pachet. Numărul de ordine al primului octet din oricare mesaj este 1.

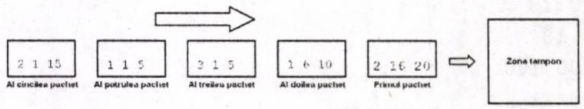


Figura 7

De exemplu, în figura 7 sunt prezentate trei mesaje (cu dimensiunile de 10, 20, respectiv 5 octeți) și cinci pachete. Dimensiunea minimă a zonei tampon pentru acest exemplu este de 10 octeți. Pe măsură ce sosesc, primul și al doilea pachet sunt stocate în zona tampon; acestea ocupă zece octeți. Urmează al treilea pachet care trece de zona tampon imediat. Al patrulea pachet trece și el imediat de zona tampon și apoi al doilea pachet trece de zona tampon. În sfârșit al cincilea pachet trece direct de zona tampon, după care trece de zona tampon și primul pachet.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **network.in**.

Un set de date începe cu o linie pe care se află două numere întregi N ($1 \le N \le 5$) și M ($1 \le M \le 1000$).

Cea de-a doua linie conține N numere întregi care reprezintă dimensiunile mesajelor, exprimate în octeți. Primul număr reprezintă dimensiunea primului mesaj, cel de-al doilea număr reprezintă dimensiunea celui de-al doilea mesaj și așa mai departe.

Fiecare dintre următoarele M linii descriu câte un pachet prin trei numere întregi: numărul de ordine al mesajului, poziția în mesaj a primului octet din pachet și poziția în mesaj a ultimului octet din pachet.

Niciunul dintre pachete nu poate conține mai mult de 64 de octeți.

Ultimul set de date este urmat de o linie pe care se află două zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de dimensiunea minimă a zonei tampon.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

network.in

```
3 3
5 5 5
1 1 5
2 1 5
3 1 5
```

```
3 5
10 20 5
2 16 20
1 6 10
3 1 5
1 1 5
2 1 15
0 0
```

Ieșire

```
Case 1: 0

Case 2: 10
```

P070322: Acoperiș

V-ați angajat să ajutați un specialist în acoperișuri să determine cantitatea de material necesară pentru a proteja acoperișului de deteriorările provocate de soare. Pentru aceasta va trebui să determinați suprafața totală a acoperișului care este vizibilă dacă este privită direct de deasupra.

Geometria unui acoperiș este dată de vârfurile triunghiurilor care cuprind suprafețele plane ale acoperișului. Coordonatele x și y ale unui punct sunt determinate pe baza privirii de deasupra, iar coordonata z reprezintă înălțimea punctului față de pământ, presupunând că suprafața pe care este amplasată clădirea este plană.

De exemplu, un segment de dreaptă delimitat de vârfurile cu coordonatele (10, 10, 10) și (10, 20, 10) este paralel cu solul și are lungimea egală cu 10 unități. Un segment delimitat de vârfurile cu coordonatele (10, 10, 10) și (10, 18, 16) urcă de la o înălțime de 10 unități la o înălțime de 16 unități deasupra solului; văzut de sus, segmentul pare a avea o lungime de 8 unități, dar lungimea reală este de 10 unități.

Din perspectiva aeriană, este posibil ca o regiune a acoperișului să acopere o altă regiune aflată mai jos. Doar porțiunile vizibile trebuie incluse în total.

De exemplu, să presupunem că privim o clădire cu două niveluri, cu un al doilea nivel care este mai mic decât primul. Evident, întreaga suprafață corespunzătoare celui de-al doilea nivel va trebui inclusă, dar suprafața vizibilă a acoperișului de la primul nivel nu va include suprafața aflată direct sub acoperișul de la al doilea nivel.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text `roof.in`.

Un set de date începe cu o linie pe care se află două numere întregi V ($1 \le V \le 300$) și T ($1 \le T \le 1000$). Valoarea V indică numărul vârfurilor, iar valoarea T indică numărul triunghiurilor.

Fiecare dintre următoarele V linii conține coordonatele x, y și z ale unui vârf.

Descrierea setului de date se încheie cu cele T triunghiuri, fiecare dintre acestea fiind caracterizat prin numerele de ordine a trei dintre vârfuri.

Vârfurile sunt numerotate în ordinea în care sunt descrise, începând de la 1.

Toate coordonatele sunt numere întregi pozitive, cel mult egale cu 100.

Niciunul dintre triunghiurile care descriu acoperișul nu este degenerat și toate triunghiurile au interioarele disjuncte (se pot atinge, dar nu se pot intersecta sau suprapune).

Ultimul set de date este urmat de o linie pe care se află două zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de suprafața vizibilă a acoperișului, tipărită cu o precizie de două zecimale.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

`roof.in`

```
6 2
10 10 10
10 20 10
20 10 10
10 10 20
10 20 20
20 20 20
1 2 3
4 5 6
3 1
10 10 10
10 18 16
20 10 10
1 2 3
0 0
```

Ieșire

```
Case 1: 75.00

Case 2: 50.00
```

P070323: Turnuri de apă

Compania *Aqua Container Management* se ocupă cu stocarea apei. Ei se gândesc la un sistem pentru păstrarea apei care constă din mai multe turnuri de apă conectate.

Fiecare turn are o secțiune orizontală cu suprafața de un metru pătrat, dar turnurile au înălțimi diferite.

Baza fiecărui turn se află pe sol și fiecare turn este legat de turnul următor printr-o conductă orizontală.

Conductele care leagă turnurile sunt orizontale, dar se află la înălțimi din ce în ce mai mari; așadar, conducta care leagă cel de-al i - lea turn de cel de-al $(i + 1)$ - lea turn se află la o înălțime mai mare decât cea care leagă cel de-al i - lea turn de cel de-al $(i - 1)$ - lea turn.

Primul turn (identificat prin 1) este deschis, astfel încât apa și aerul pot intra liber în el, prin partea superioară. Toate celelalte turnuri sunt închise, deci apa și aerul pot intra doar prin conductele de legătură.

Aceste conducte sunt suficient de mari astfel încât apa și aerul să poată circula liber și simultan prin ele, dar suficient de mici pentru ca dimensiunile lor să poată fi ignorate pentru problema de față.

Turnurile sunt umplute prin introducerea lentă a apei în primul turn, până în momentul în care acesta este umplut. Pe măsură ce nivelul apei crește deasupra conductelor de legătură, apa curge prin celelalte turnuri.

Aqua Container Management needs are nevoie de un program care calculează numărul de metri cubi care pot fi introduși în turnuri înainte ca primul turn să fie umplut.

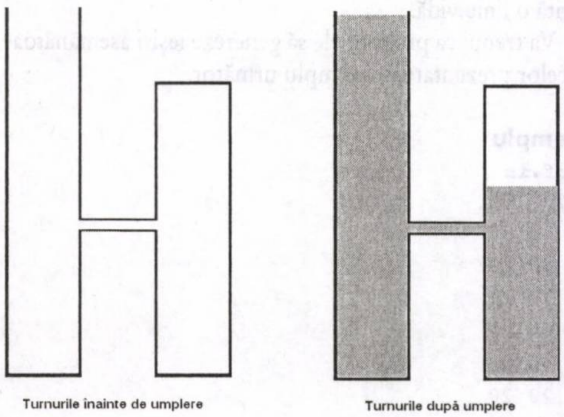


Figura 8

În figura 8 este prezentat un exemplu simplu care constă în numai două turnuri. După ce procedura de umplere se încheie, aerul din partea superioară a celui de-al doilea turn este comprimat (presiunea sa este mai mare decât o atmosferă), deci nivelul apei din al doilea turn este mai scăzut decât nivelul apei din primul turn.

Următoarele principii fizice sunt utile în rezolvarea acestei probleme (unele dintre principii sunt aproximări acceptabile pentru scoprile acestei probleme):

- apa curge în jos;
- în spațiul liber, presiunea aerului este de o atmosferă;
- aerul este compresibil (volumul ocupat de o anumită cantitate de aer depinde de presiune); apa nu este compresibilă (volumul ocupat de o anumită cantitate de apă este constant, indiferent de presiune);
- într-un spațiu închis, presiunea aerului este aceeași; dacă volumul spațiului liber se modifică, produsul dintre volumul de aer și presiunea acestuia rămâne constant; de exemplu, să presupunem că într-un spațiu închis volumul inițial este V_1, iar presiunea este P_1; dacă volumul devine V_1, atunci noua presiune P_2 satisface relația $P_1 \cdot V_1 = P_2 \cdot V_2$.
- într-o coloană de apă aflată sub un spațiu ocupat de aer, presiunea apei la un nivel de D metri sub suprafața apei este dată de presiunea aerului la care se adaugă $0{,}097 \cdot D$ atmosfere; această formulă este valabilă indiferent dacă spațiul ocupat de aer este închis sau deschis;
- într-un spațiu ocupat de apă (de exemplu, dacă două sau mai multe turnuri sunt conectate prin conducte aflate sub nivelul apei, presiunea apei este constantă la oricare nivel.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **tanks.in**. Fiecare set de date constă în trei linii.

Prima dintre aceste linii conține un număr întreg N ($2 \le N \le 10$) care reprezintă numărul turnurilor.

Cea de-a doua linie conține N numere reale, care reprezintă înălțimile, exprimate în metri, ale turnurilor, în ordinea numerelor de ordine ale acestora.

Cea de-a treia linie conține $N - 1$ numere reale, care reprezintă înălțimile, exprimate în metri, la care se află conductele care leagă turnurile. Aceste numere formează o secvență crescătoare (fiecare dintr ele este mai mare decât predecesorul său).

Ultimul set de date este urmat de o linie pe care se află doar un zero.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de cantitatea de apă, exprimată în metri cubi, care poate fi introdusă înainte ca nivelul apei să ajungă în partea superioară a primului turn. Această cantitate va fi scrisă cu o precizie de trei zecimale.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă.

Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

tanks.in

```
2
10.0 8.0
4.0
0
```

Ieșire

```
Case 1: 15.260
```

P070324: Tuneluri

Un spion chipeș a scăpat din elaborata capcană mortală pe care i-ați întins-o, v-a învins gardieni și a scăpat cu planurile de dominare mondială pe care le-ați pus la punct.

Acum el se plimbă prin baza dumneavoastră din vulcan și întreaga operație malefică este supusă unui risc. Trebuie oprit înainte să scape!

Din fericire, puteți să urmăriți progresul spionului din camera de control secretă și aveați planuri pentru o astfel de eventualitate.

Baza dumneavoastră constă într-o rețea complicată de camere legate între ele prin tuneluri care nu se intersectea-

ză. Fiecare încăpere conține o cameră video care vă permite să îl urmăriți pe spion oriunde s-ar duce și fiecare tunel conține o cantitate mică de combustibil, suficient de puternică pentru a provoca colapsul tunelului.

Spionul este prea rapid pentru a fi prins atunci când un tunel se prăbușește, deci va trebui să prăbușiți tunelurile în mod strategic pentru a nu-i permite să parcurgă drumul din camera în care se afla inițial până în afara bazei.

Daunele provocate bazei vor necesita cheltuieli costisitoare pentru ca totul să revină la normalitate, deci ați dori să prăbușiți cât mai puține tuneluri.

Găsiți o strategie care minimizează numărul tunelurilor care trebuie prăbușite pentru ca spionul să nu poată scăpa indiferent cât de inteligent este.

Pentru a nu avea surprize, va trebui să presupuneți că spionul știe totul despre sistemul de tuneluri al bazei. Avantajul principal pe care îl aveți este că puteți să prăbușiți tuneluri oricând doriți, pe baza observațiilor pe care le faceți pe măsură ce spionul se deplasează prin tuneluri.

Date de intrare

Programul va citi mai multe seturi de date din fișierul text **tunnels.in**.

Prima linie corespunzătoare unui set de date numerele întregi R ($1 \le R \le 50$) și T ($1 \le T \le 1000$), care reprezintă numărul camerelor, respectiv numărul tunelurilor. camerele sunt numerotate de la 1 la T.

Urmează T linii care conțin, fiecare, câte două numere întregi x și y ($0 \le x, y \le R$), care indică numerele de ordine a două camere legate printr-un tunel; valoarea 0 indică faptul că tunelul face legătura cu exteriorul. Două camere pot fi legate direct prin mai multe tuneluri.

Spionul va porni întotdeauna din camera cu numărul de ordine 1.

Ultimul set de date este urmat de o linie pe care se află două zerouri.

Date de ieșire

Pentru fiecare set de date se va scrie la ieșirea standard o linie care începe cu numărul de ordine al testului (numerotarea începe de la 1), urmat de numărul tunelurilor care trebuie prăbușite în cel mai defavorabil caz.

După linia corespunzătoare fiecărui set de date va fi tipărită o linie vidă. Va trebui ca programele să genereze ieșiri asemănătoare celor prezentate în exemplu următor.

Exemplu

tunnels.in

```
4 6
1 2
1 3
2 4
3 4
4 0
4 0
4 6
1 2
1 3
1 4
2 0
3 0
4 0
0 0
```

Ieșire

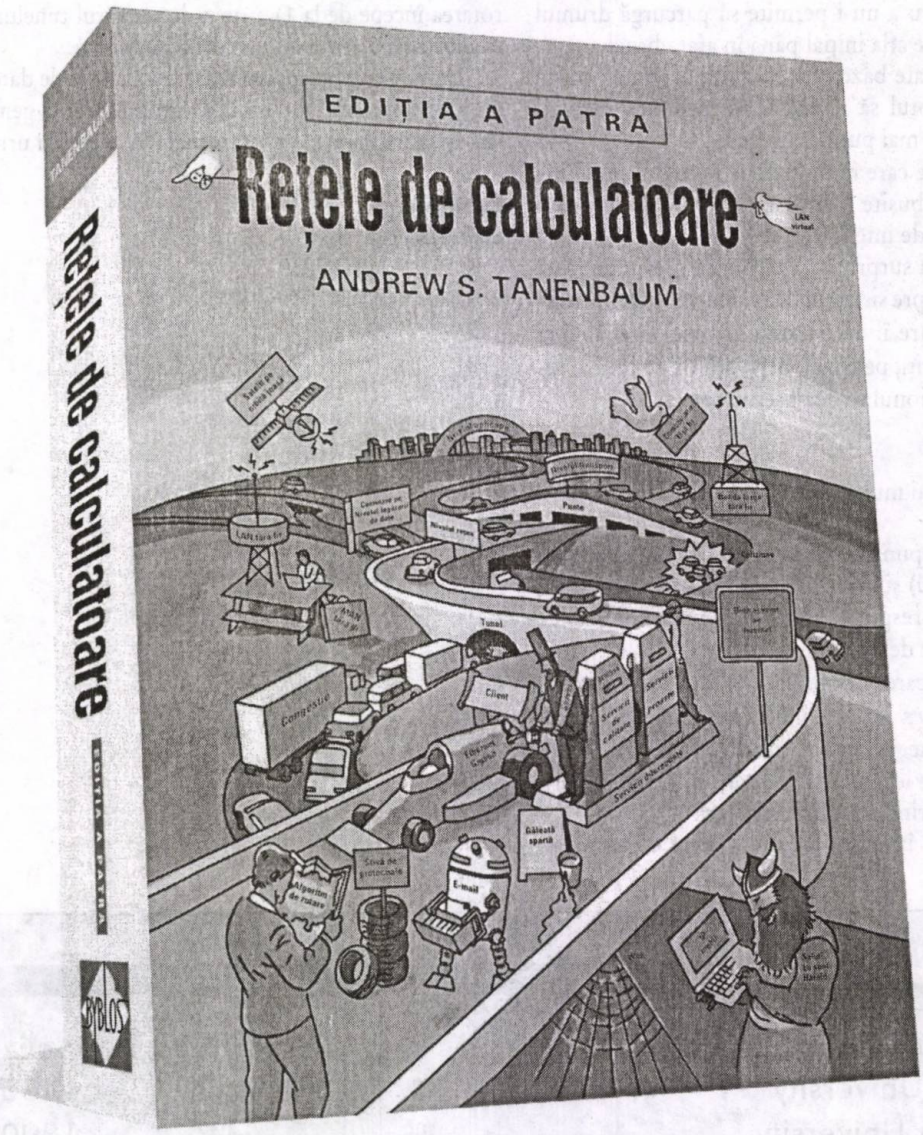
```
Case 1: 2

Case 2: 2
```

Clasament

Echipă	Probleme	Penalizare
1. Warsaw University	8	1405
2. Tsinghua University	7	1200
3. St. Petersburg University of IT, Mechanics and Optics	6	866
4. Massachusetts Institute of Technology	6	866
5. Novosibirsk State University	6	868
6. Saratov State University	6	957
7. Twente University	6	1011
8. Shanghai Jiao Tong University	6	1026
9. University of Waterloo	6	1103
10. Moscow State University	6	1192
11. University of Auckland	6	1210
12. California Institute of Technology	6	1241

Cine are carte, are parte!

...de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețelele de calculatoare

- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o REDUCERE semnificativă!

Byblos srl, tel: 0318-056-314, 0721-927-499, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librărie: Galeria NOI-La Calu' Bălan, Str. Halelor Nr. 3 (după Hanul lui Manuc).

Aplicații ale proprietăților de DIVIZIBILITATE

Lucian Ilea, Iulian Pah

Scopul articolului de față este de a ilustra diverse proprietăți ale numerelor naturale, legate de numărul divizorilor și de unele descompuneri ale acestora.

Problema #1: Descompuneri

Se consideră un număr natural x, nenul, $1 \le x \le 1000000$. Se cere să se determine numărul de scrieri ale lui x ca sumă de numere naturale nenule consecutive.

Exemplu

Pentru $x = 9$ avem trei descompuneri, și anume:

- $9 = 9$;
- $9 = 4 + 5$;
- $9 = 2 + 3 + 4 + 5$.

Soluție

O demonstrație matematică accesibilă elevilor de liceu va conduce la o concluzie simplă, anume că, pentru un x oarecare, numărul de posibilități de descompunere ca sumă de numere naturale consecutive este egal cu numărul divizorilor săi impari.

Deoarece $1 + 2 + ... + x = x \cdot (x + 1) / 2$ înseamnă că trebuie determinat numărul soluțiilor în mulțimea numerelor naturale ale ecuației $x = n \cdot (n + 1) / 2 - m \cdot (m + 1) / 2$.

Rezolvând aceasta ca o ecuație de grad 2 în m se obține $m = -1 + [1 - 4 \cdot (2 \cdot x - n \cdot (n + 1))]^{1/2} / 2$.

Numărul de sub radical (cel ridicat la puterea 1/2) trebuie să fie pătratul unui număr natural impar, notat în continuare cu u.

Se obține egalitatea $4 \cdot n^2 + 4 \cdot n + 1 - 8 \cdot x - u^2 = 0$, care, rezolvată în necunoscuta n, conduce la $n = -1 + (8 \cdot x + u^2)^{1/2} / 2$.

Cu aceeași condiție referitoare la valoarea de sub radical se obține $8 \cdot x = u^2 - v^2$, adică $8 \cdot x = (u - v) \cdot (u + v)$.

Ecuația se poate rescrie sub forma $2 \cdot x = [(u - v) / 2] \cdot [(u + v) / 2]$. Se notează primul factor cu A și al doilea cu B și se constată că A și B trebuie să aibă parități diferite, deoarece $v = A + B$ și $u = B - A$.

Este evident că orice pereche (A, B) de numere cu proprietatea $2 \cdot x = A \cdot B$ și cu exact unul dintre A și B număr impar conduce la o modalitate de a rezolva ecuația inițială și, implicit, la o descompunere ca sumă de numere consecutive pentru x.

Problema #2: Dispozitive

Se consideră n dispozitive electrice care sunt numerotate cu numere naturale consecutive, începând de la 1.

Dispozitivele pot avea două stări (0 sau 1). Starea unui dispozitiv se poate schimba alternativ (din 0 devine 1 și din 1 devine 0). Inițial toate dispozitivele se află în starea 0.

La pasul 1 toate dispozitivele își schimbă starea. La pasul 2 dispozitivele 2, 4, 6, ... își schimbă starea. La pasul k dispozitivele k, $2 \cdot k$, $3 \cdot k$, ... își schimbă starea.

În total se efectuează n pași (numărul de pași este egal cu numărul de dispozitive).

Se cere să se determine numărul dispozitivelor care în final se găsesc în starea 1. Se știe că $1 \le n \le 1000000000$.

Soluție

Cele n dispoxitive formează un șir X. Evident, o încercare de a rezolva problema prin efectuarea operațiilor și numărarea finală a valorilor 1 din șirul X este prea costisitoare în ceea ce privește timpul de execuție, dar studiul comportamentului valorilor din X va conduce la formularea unor

observații importante. Considerăm $n = 6$. În acest caz și răspunsul este 2.

pas	*X*
0	000000
1	111111
2	101010
3	100011
4	100111
5	100101
6	100100

Se observă ușor că starea dispozitivului k este influențată numai de dispozitivele ale căror numere de ordine sunt divizori ai lui k.

Deoarece stările alternează, o primă reformulare a problemei ar fi să se determine numărul de numere de la 1 la n care au un număr impar de divizori.

Deoarece singurele numere care au un număr impar de divizori sunt pătratele perfecte, o a doua reformulare a problemei cere să se determine numărul de pătrate perfecte din intervalul $[1, n]$, cu răspunsul final $[n^{1/2}]$.

Problema #3: Număr de întrebări

Cineva se gândește la un număr rațional, fracție ireductibilă) strict pozitiv și subunitar, cu numitorul mai mic sau egal decât n (număr natural mai mare decât 1). Se notează acest număr cu x.

Se pot pune mai multe întrebări de forma: "este adevărat că $x < a$?", unde a este un număr real oarecare. Scopul "jocului" este de a ghici fracția x, punând un număr minim de întrebări.

Se cere, ca pentru un număr n oarecare, să se determine un număr minim de întrebări suficient pentru a determina orice fracție x care îndeplinește proprietățile menționate mai sus. Se știe că $1 < n \leq 5000$.

Exemplu

Pentru $n = 4$, fracțiile ireductibile, strict pozitive și subunitare cu numitorul mai mic sau egal cu 4 sunt: 1/4, 1/3, 1/2, 2/3, 3/4, iar numărul minim de întrebări cu care se poate face determinarea este 3.

Soluție

Secvența fracțiilor ireductibile strict pozitive și subunitare, cu numitorul mai mic decât n, scrise în ordine crescătoare, se numește *secvență Farey de ordin n* și se notează prin F_n.

De exemplu, avem $F_4 = \{1/4, 1/3, 1/2, 2/3, 3/4\}$ sau $F_5 = \{1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5\}$.

Este evident că numărul de întrebări cerut este partea întreagă superioară a valorii $\log_2(|F_n|)$ ($|A|$ este cardinalul unei mulțimi A).

Pentru determinarea valorii $|F_n|$ se observă că F_n conține toate elementele lui F_{n-1} și, în plus, toate fracțiile ireductibile cu numitor n.

Se obține $|F_n| = |F_{n-1}| + \varphi(n)$, unde $\varphi(n)$ este funcția indicator a lui *Euler*, $\varphi(n)$ fiind numărul valorilor naturale pozitive, mai mici decât n și prime cu n.

Se demonstrează ușor că funcția φ are două proprietăți importante:

- dacă p este un număr prim și k este un număr natural, atunci $\varphi(p^k)=(p - 1) \cdot p^{k-1}$;
- dacă p și q sunt prime între ele, atunci $\varphi(p \cdot q)= \varphi(p) \cdot \varphi(q)$.

De aici reiese că, dacă n este descompus în factori primi, $n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$, atunci $\varphi(n) = (p_1 - 1) \cdot p_1^{e_1-1} \cdot (p_2 - 1) \cdot p_2^{e_2-1} \cdot \ldots (p_k - 1) \cdot p_k^{e_k-1}$, ceea ce furnizează o rezolvare eficientă a problemei.

Bibliografie

1. http://acm.mipt.ru;
2. http://infoarena.ro.

Vulnerabilitate

Atacurile diverșilor glumeți nu puteau cruța site-urile unor instituții publice din România (care, se pare, sunt protejate în "dulcele stil românesc").

Astfel, la sfârșitul anului trecut, timp de aproape o zi, site-ul oficial al Guvernului României, precum și cel al Ministerului Afacerilor Externe ne informau: "Cunoscutul om de afaceri și fotbal George Becali, președinte al Partidului Noua Generație, a fost numit astăzi în funcția de ministru de Externe, imediat după ce ministrul în funcție, Răzvan Ungureanu, și-a anunțat demisia".

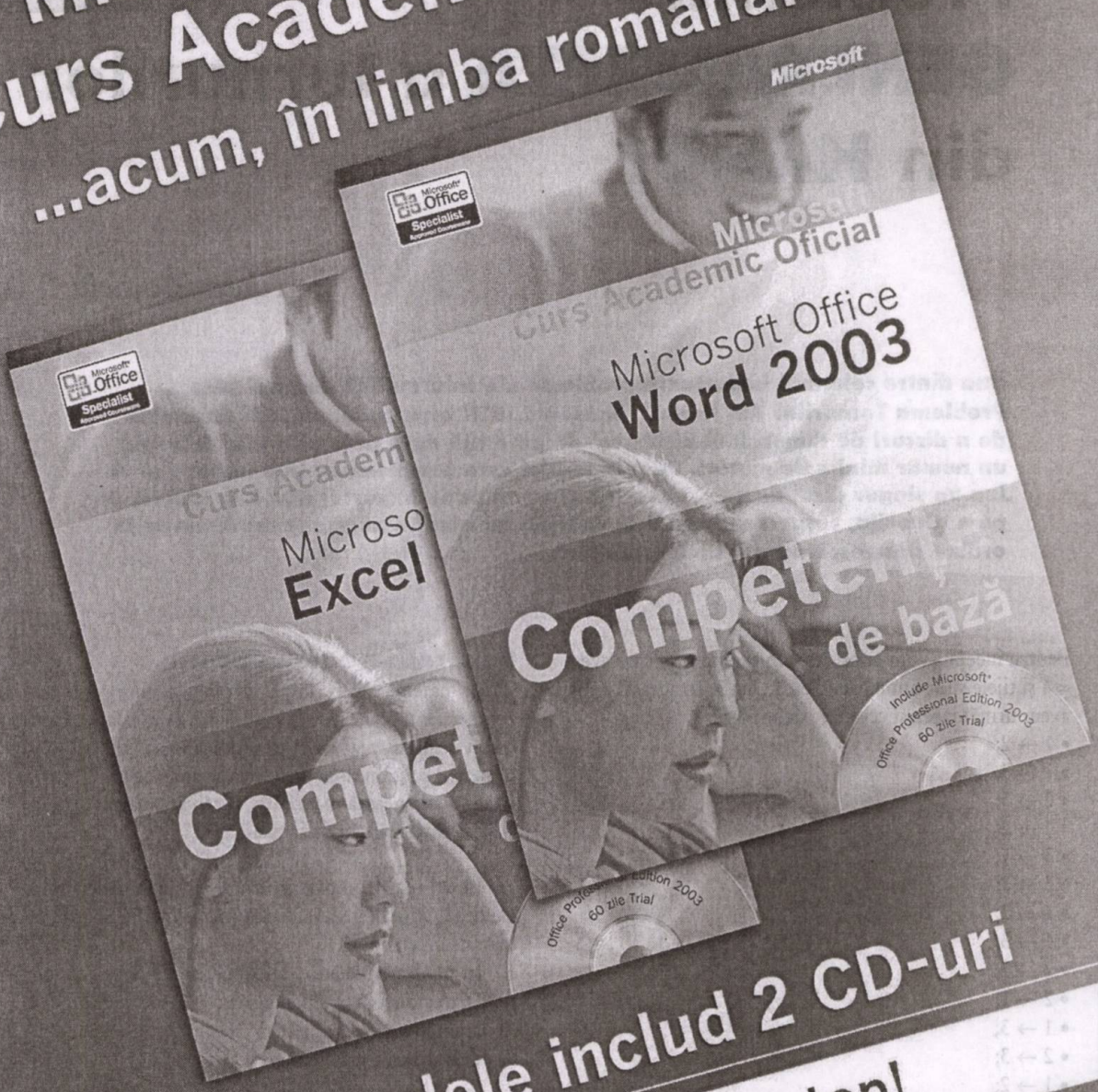
Mai mult, ar fi putut părea că nici președintele țării nu o duce prea bine. Aceleași site-uri "susțineau": "Cunoscutul om politic și de afaceri Dan Voiculescu, președinte al Partidului Conservator, s-a autodeclarat Președinte al României, imediat ce lovitura de stat fulger pe care a inițiat-o a avut un succes neașteptat".

Evident, nimeni nu a luat în serios astfel de informații, dar faptul că cineva se poate juca pe aceste site-uri ar putea fi îngrijorător.

Poate ar trebui să ne gândim că nu toți specialiștii români din domeniu sunt atât de buni pe cât ne-ar plăcea să credem.

NOU!

Microsoft®
Curs Academic Oficial
...acum, în limba română!



Manualele includ 2 CD-uri

Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Problema GENERALIZATĂ a turnurilor din HANOI

Tiberiu Socaciu

Una dintre cele mai importante probleme ale informaticii de bază este Problema Turnurilor din Hanoi[1]. În esență, PTH cere mutarea unui ansamblu de n discuri de dimensiuni distincte[2] de pe o tijă dată pe o altă tijă[3] folosind un număr minim de mutări. Regula jocului este următoarea: la o mutare se va lua un singur disc[4] de pe oricare tijă și se va pune în vârful stivei de discuri de pe o altă tijă, respectând condiția ca tijele să aibă în permanență discurile în ordine descrescătoare a dimensiunilor[5].

Pentru început vom considera un exemplu în care avem $n = 4$ și tijele sunt numerotate cu 1, 2 și 3. În această situație avem următoarea succesiune de mutări:

- $1 \rightarrow 2$;
- $1 \rightarrow 3$;
- $2 \rightarrow 3$;
- $1 \rightarrow 2$;
- $3 \rightarrow 1$;
- $3 \rightarrow 2$;
- $1 \rightarrow 2$;
- $1 \rightarrow 3$;
- $2 \rightarrow 3$;
- $2 \rightarrow 1$;
- $1 \rightarrow 3$;
- $2 \rightarrow 3$;
- $1 \rightarrow 2$;
- $1 \rightarrow 3$;
- $2 \rightarrow 3$.

Se poate demonstra relativ ușor că numărul minim de mutări este $2^n - 1$.

[1] *vom prescurta cu PTH*
[2] *așezate descrescător, după dimensiune, de la bază la vârf*
[3] *având la dispoziție o tijă de manevră*
[4] *cel din vârf*
[5] *de la bază spre vârf*

Generalizare și extindere

Orice matematician[6], după rezolvarea unei probleme de geometrie în plan trece la extinderea ei în spațiu. Evident, există posibilitatea[7] ca, la extinderea în spațiu:
a) să nu se întâlnească nicio relație analoagă;
b) să se întâlnească mai multe relații analoage;
c) să se întâlnească relația analoagă.

În cel mai frumos caz, cazul *c*), matematicianul se și gândește apoi la generalizarea problemei în spațiul euclidian n-dimensional.

În mod asemănător, informaticienii[8] încearcă, pentru rezolvarea unei probleme banale:
a) parametrizarea problemei;
b) generalizarea problemei.

Acest lucru se realizează în speranța măririi spațiului problemelor rezolvabile cu ajutorul respectivului cod scris. Astfel, în cazul punctului *a*) sunt acum la moda tehnicile orientate obiect[9] de definire a ierarhiilor de clase bazate pe moștenire precum și un puternic curent în favoarea reutilizării *soft*-ului.

[6] *chiar și cel în devenire*
[7] *în funcție de problema inițială*
[8] *chiar și cei în devenire*
[9] *Object Oriented, OO*

Generalizarea PTH

Într-o tabără de informatică[10] am susținut un material unde am definit pentru prima dată *problema generalizată a turnurilor din Hanoi*[11], regulile și scopul find aceleași ca la *PTH*. Dacă notăm cu $m + 2$ numărul total de tije[12] avem, în mod evident, $PGTH(1) = PTH$.

Dacă vom nota cu $p(m, n)$ numărul minim de mutări necesare rezolvării problemei $PGTH(m)$ atunci, conform principiului optimalității[13] al lui *Bellman*, avem:

- $R_1(m)$: $p(m, 0) = 0$;
- $R_2(m)$: $p(m, n + 1) = \min \{1 + 2 \cdot (p(m, x_1) + p(m - 1, x_2) + ... + p(m - k + 1, x_k) \mid x$ este o partiție de lungime m a lui $n\}$.

Cu alte cuvinte, pentru a muta[14] ansamblul de $n + 1$ discuri când avem la dispoziție m tije intermediare libere, trebuie întâi să determinăm o partiție $(x_1, ..., x_k)$ de lungime m a lui n, după care vom muta (optim) x_1 discuri având la dispoziție un număr de $m - 1$ tije intermediare, apoi cele x_2 discuri având la dispozitie $m - 2$ tije etc.; apoi, vom muta cel de-al $(n + 1)$ - lea disc[15] direct pe tija destinație, următoarele etape fiind mutarea discurilor de pe tijele intermediare pe tija destinație în ordinea inversă a mutării lor de pe tija sursă. Determinarea partiției se va realiza astfel încât să se minimizeze numărul de mutări.

[10] *vezi [1]*
[11] *PGTH*
[12] *m este un număr natural nenul*
[13] *vezi [2]*
[14] *optim*
[15] *cel mai mare*

Extinderea PGTH (cazul m=2)

Extinderea problemei în cazul cu patru tije ($m = 2$) a fost prezentată împreună cu modul de rezolvare în [1], după care a fost publicată în [2][16].

Astfel dacă notăm $q(n) = p(2, n)$, pentru orice număr natural n, atunci recurența are soluția $q(n) = (n - a \cdot (a + 3) / 2)$, unde $a = [(1 + (1 + 8 \cdot n)^{1/2}) / 2]$.

Determinarea formulei și a strategiei

Pentru a "ghici" proprietățile valorilor p vom folosi programul **PGTH1.PAS**[17] care determină câteva elemente[18]. Se observă că pentru un m fixat, diferența între doi termeni consecutivi din șirul $p(m, \cdot)$ este[19] o putere a lui 2.

Vom nota $r(m, n) = p(m, n) - p(m, n - 1)$, pentru oricare numere naturale nenule m și n.

Dacă încercăm să numărăm elementele consecutive de aceeași valoare ale acestor diferențe, observăm că avem $A(i, j) = (i + j - 2)! / ((i - 1)! (j - 1)!)$.

$A(m, n)$ reprezintă numărul elementelor[20] șirului $r(m, \cdot)$ care sunt egale cu 2^n. Se pot demonstra relativ ușor următoarele proprietăți ale funcției A:

1. $A(m, n) = A(n, m)$, pentru oricare două numere naturale nenule m și n;
2. $A(m, n - 1, n) = A(m, n)$, pentru oricare două numere naturale nenule m și n;

[16] *pentru rezolvarea recurenței R(2) au fost enunțate și demonstrate un număr de șase leme și o teoremă*
[17] *vezi listing-ul*
[18] *vezi tabelarea funcției p*
[19] *vezi tabelarea funcției r*
[20] *vezi tabelarea funcției A*

Valorile p

n	p(1, n)	p(2, n)	p(3, n)	p(4, n)	...
0	0	0	0	0	...
1	1	1	1	1	...
2	3	3	3	3	...
3	7	5	5	5	...
4	15	9	7	7	...
5	31	13	11	9	...
6	63	17	15	13	...
7	127	25	19	17	...
8	255	33	23	21	...
9	511	41	27	25	...
10	1023	49	31	29	...
11	2047	65	39	33	...
12	4095	81	47	37	...
13	8191	97	55	41	...
14	16383	113	63	45	...
15	32767	129	71	49	...
...	...	...	...	...	...

Valorile r

n	r(1, n)	r(2, n)	r(3, n)	r(4, n)	...
1	1	1	1	1	...
2	2	2	2	2	...
3	4	2	2	2	...
4	8	4	2	2	...
5	16	4	4	2	...
6	32	4	4	4	...
7	64	8	4	4	...
8	128	8	4	4	...
9	256	8	4	4	...
10	512	8	4	4	...
11	1024	16	8	4	...
12	2048	16	8	4	...
13	4096	16	8	4	...
14	8192	16	8	4	...
15	16384	16	8	4	...
...	...	...	...	...	...

Valorile A

i / j	1	2	4	5	6	...
1	1	1	1	1	1	...
2	1	2	3	4	5	6	...
3	1	3	6	10	15	21	...
4	1	4	10	20	35	56	...
5	1	5	15	35	70	126	...
6	1	6	21	56	126	252	...
...	...	...	...	...	...	...	...

3. $A(1, n) + A(2, n) + ... + A(m, n) = A(m, n + 1)$, pentru oricare două numere naturale nenule m și n;
4. funcția $f_n: \mathbf{N} \setminus \{0\} \to \mathbf{N}$, dată de $f_n(m) = A(m, n)$, pentru oricare două numere naturale nenule m și n, este o funcție crescătoare;
5. $s(m, n)$, definit ca find acel număr care verifică $A(s(m, n), n) \le m < A(s(m, n) + 1, n)$ este bine definit.

În aceste condiții, mai rămâne de arătat că funcția $p(\cdot, \cdot)$, definită prin dubla recurență $R_1(m)$ și $R_2(m)$ este definită prin:

$$p(m, n) = A(m, 1) \cdot 2^{1-1} + (n - A(m, s(m + 1, n) + 1)) \cdot 2^1 + A(m, 2) \cdot 2^{2-1} + (n - A(m, s(m + 1, n) + 1)) \cdot 2^2 + ... + A(m, s(m + 1, n) - 1) \cdot 2^{s(m+1,n)-1-1} + (n - A(m, s(m + 1, n) + 1)) \cdot 2^{s(m+1,n)-1}$$

Cu alte cuvinte, rezultatul este dat de suma termenilor de forma $A(m, k) \cdot 2^{k-1} + (n - A(m, s(m + 1, n) + 1)) \cdot 2^k$, unde k variază între 1 și $s(m + 1, n) - 1$.

Evident, o formulă de acest tip este mai utilă decât dubla recurență $R_1(m)$ și $R_2(m)$; astfel, printr-o simplă iterație se determină valoarea elementului $p(m, n)$, spre deosebire de determinarea tuturor partițiilor lui n, care necesită un algoritm de ordin exponențial[21]. Avantajele folosirii acestei formule față de recursivitatea $R_1(m)$ și $R_2(m)$ se pot observa studiind *listing*-ul programului **PGTH2.PAS**.

Bibliografie

1. Tiberiu Socaciu, *Generalizarea problemei turnurilor din Hanoi*, comunicare la Tabăra "Așii informaticii", Piatra Neamț, iunie 1995;
2. Tiberiu Socaciu, *O extindere a problemei Turnurilor din Hanoi*, Gazeta de Informatică nr. 11 - 12 / 1994, pp. 19-27, Editura Libris Cluj-Napoca;
3. Tiberiu Socaciu, *Rezolvarea unei probleme de Backtracking*, GInfo 17/1 (ianuarie 2007), Editura Agora Media Târgu Mureș;
4. Tiberiu Socaciu, *Învățați Backtracking. Exemple în Pascal și C*, Editura InfoData Cluj-Napoca, 2006;
5. Tiberiu Socaciu, *Metoda Backtracking cu exemple în Pascal*, Editura Edusoft Bacău, 2005;
6. Tiberiu Socaciu, *Metoda Backtracking cu exemple în C*, Editura Edusoft Bacău, 2005.

[21] *vezi [3], [4], [5] și [6]*

Listing PGTH1.PAS (I)

```
Program pgth1;
const maxm = 6;
const maxn = 20;
var p: array[0..maxn, 1..maxm] of Longint;
var i, n, m: Integer;

procedure gen(n, m:Integer);
  var x: array[1..maxm] of Integer;
  var mm: Integer;
  var minim: Longint;

  procedure prelucrare;
    var k: Integer;
    var s: Longint;
  begin
    s := 0
    for k := 1 to mm do s: =s + p[x[k], k];
    if minim < 0 then minim := s
    else
      if s < minim then minim := s
    end;

  procedure genereaza(n, m: Integer);
    var i: Integer;
  begin
    if m = 1 then
  begin
      x[1] := n;
      prelucrare
    end
```

Listing PGTH1.PAS (II)

```
    else
      for i := 0 to n do
      begin
        x[m] := n - i;
        genereaza(i, m - 1)
      end
  end;

begin
  mm := m;
  minim := -1;
  genereaza(n - 1, m);
  p[n, m] := 1 + 2 * minim
end;

Begin
  for i := 1 to maxm do p[0,i] := 0;
  for n := 1 to maxn do
  begin
    for m := 1 to maxm do
    begin
      gen(n, m);
      Write(p[n, m]:10)
    end;
    Writeln
  end
End.
```

Listing PGTH2.PAS (I)

```
Program pgth2;
const maxn = 20;
const maxm = 6;
var a: array[1..maxn, 1..maxm + 1] of Longint;
var m, n: Integer;

procedure init_a;
begin
  for m := 1 to maxm + 1 do
    a[1, m] := 1;
  for n := 2 to maxn do
 begin
    a[n, 1] := 1;
    for m := 2 to maxm + 1 do
      a[n, m] := a[n - 1, m] + a[n, m - 1]
  end
end;

function s(m, n: Integer): Integer;
  var k: Integer;
begin
  k := 1;
  while a[k, n] <= m do
    Inc(k);
  s := k - 1
End;

function p(n, m: Integer): Longint;
  var k, ss: Integer;
  var putere, suma: Longint;
begin
  suma := 0;
  putere := 1;
  ss := s(n, m + 1);
  for k := 1 to ss do
  begin
    suma := suma + putere * a[k, m];
    putere := putere * 2
  end;
  suma := suma + (n - a[ss, m + 1]) * putere;
  p := suma
end;

Begin
  init_a;
  for n := 1 to maxn do
  begin
    for m := 1 to maxm do
      Write(p(n, m):11);
    Writeln
 end
End.
```

Știați că...

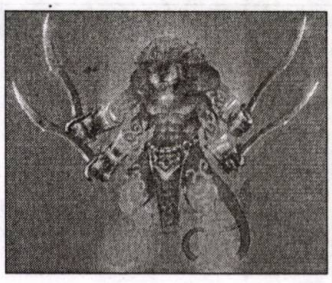
Problema turnurilor din Hanoi apare, sub diverse forme, în numeroase jocuri video. Datorită simplității în implementare și a ușurinței cu care poate fi recunoscută, ea apare ca puzzle în numeroase jocuri de aventură. Câteva dintre aceste jocuri sunt:

- **Black & White;**
- **The Island of Dr. Brain;**
- **The Secret Island of Dr. Quandary;**
- **Star Wars: Knights of the Old Republic;**
- **Zork Zero;**
- **The Legend of Kyrandia: Hand of Fate.**

Problema apare și în mai multe romane sau nuvele SF. Un exemplu interesant este **Now Inhale** de **Eric Frank Russell**. Eroul este capturat și, pentru a i se cruța viața, trebuie să aleagă un joc pe care să îl câștige. Condiția impusă este ca jocul să poată avea loc în celulă și să se poată încheia după un număr finit de pași. Știind că o navă de salvare va sosi peste aproximativ un an, scopul prizonierului este ca jocul să dureze cât mai mult; din acest motiv, el lege problema turnurilor din Hanoi cu 64 de discuri.

Într-un episod al serialului **Doctor Who**, doctorului i se cere să rezolve **jocul trilogic** (problema turnurilor din Hanoi cu zece discuri), folosind exact $1023 = 2^{10} - 1$ de mutări.

Problema a fost inventată de matematicianul francez **François Édouard Anatole Lucas** (1842 - 1891). El s-a inspirat dintr-o legendă care descria un templu hindus în care o problemă similară era folosită pentru disciplinarea mintală a tinerilor preoți. Potrivit legendei, la începutul timpului preoții au primit 64 de discuri de aur, fiecare puțin mai mic decât cel de sub el. Sarcina preoților era să mute discurile de pe una dintre cele trei tije pe o alta, astfel încât niciodată un disc să nu se afle deasupra unui disc mai mare. Preoții au lucrat eficient zi și noapte. Se spunea că atunci când vor termina, templul se va dărâma și lumea va dispărea. Preoții din acest templu ar fi trebuit să efectueze nu mai puțin de $2^{64} - 1 = 18.446.744.073.709.551.615$ de mutări. Presupunând că ar fi putut fi mutat câte un disc în fiecare secundă, tot ar fi fost necesari peste 500 de miliarde de ani.



CĂUTĂRI în ȘIRURI

Alexandru Popa

În cadrul acestui articol ne propunem să prezentăm trei algoritmi de căutare a șirurilor de caractere și anume algoritmul Knuth - Morris - Pratt, algoritmul Boyer - Moore și algoritmul Rabin - Karp.

Prezentarea problemei

Fiind dat un text T și un *șablon* (engl. *pattern*) S, se cere determinarea tuturor aparițiilor șablonului S în textul T. Presupunem că textul T conține n caractere, iar șablonul S conține m caractere. De asemenea, trebuie menționat faptul că lungimea textului T poate fi foarte mare.

Algoritmul naiv

Prima idee de rezolvare este să căutăm secvențial fiecare apariție a lui S în T, adică să comparăm $S[1 .. m]$ cu $T[p .. p + m - 1]$, unde p ia valori cuprinse între 1 și $n - m + 1$. Algoritmul în pseudocod este:

```
algoritm CăutareNaivă(T, S)
  n ← lungime(T)
  m ← lungime(S)
  pentru i ← 1, n - m + 1 execută
  potrivire ← adevărat
  pentru j ← 1, m execută
    dacă T[i+j-1] ≠ S[j] atunci
      potrivire ← fals
      * ieșire forțată din ciclu
    sfârșit dacă
  sfârșit pentru
  dacă potrivire atunci
    scrie "Am găsit o potrivire cu deplasarea",
                                        i - 1
  sfârșit pentru
sfârșit algoritm
```

Complexitate

Algoritmul naiv are ordinul de complexitate $O((n - m + 1) \cdot m)$ pentru cazul cel mai defavorabil (cazul în care avem un text format din n caractere `'a'` și un șablon format din m caractere `'a'`). Dacă $m = n / 2$ atunci algoritmul are ordinul de complexitate $O(n^2)$.

Algoritmul Knuth-Morris-Pratt (KMP)

Înainte de a realiza căutarea în textul T, algoritmul *KMP* preprocesează șablonul S. Fiecărui subșir $S[1 .. i]$ i se atribuie o valoare $suprapunere_i$, care reprezintă lungimea celui mai lung prefix al lui $S[1 .. i]$, care este și sufix al acestui șir.

De exemplu, dacă avem șirul `"ababab"`, cel mai lung prefix, care este în același timp sufix al șirului dat, este `"abab"`, acesta având lungimea 4. În continuare prezentăm construcția completă a vectorului suprapunere pentru șablonul `"abcacbaba"`.

i	1	2	3	4	5	6	7	8	9
S_i	a	b	a	b	c	a	b	a	c
$suprapunere_i$	0	0	1	2	0	1	2	3	0

De ce suntem interesați să calculăm această valoare? Să presupunem că dorim să căutăm într-un text șablonul `"ababcabac"`, prezentat mai sus. De asemenea să presupunem că am reușit să potrivim primele patru caractere (`"abab"`) și al cincilea caracter este diferit de `'c'`.

Acum nu are rost să mutăm șablonul doar cu o poziție la dreapta, așa cum făceam la căutarea naivă, pentru că știm sigur că nu vom avea o potrivire (`"abab ..."` ≠ `"bab ..."`). Dacă mutăm șablonul cu două poziții la dreapta, s-ar putea să avem o potrivire.

Algoritmul în pseudocod este:

```
algoritm Preprocesare(S)
  m ← lungime(S)
  k ← 0
  suprapunere[1] ← 0
  pentru i ← 2, m execută
    cât timp k > 0 și S[k+1] ≠ S[i] execută
      k ← suprapunere[k]
    sfârșit cât timp
```

```
    dacă S_{k+1} = S_i atunci
      k ← k + 1;
    sfârșit dacă
    suprapunere_i ← k;
  sfârșit pentru
sfârșit algoritm

algoritm Căutare_KMP(T, S)
  n ← lungime(T)
  k ← 0
  Preprocesare(S)
  pentru i ← 1, n execută
    cât timp k > 0 and T_i ≠ S_{k+1}
      k ← suprapunere_k
    sfârșit cât timp
    dacă T_i = S_{k+1} atunci
      k ← k + 1;
    sfârșit dacă
    dacă k = m atunci
      scrie "Am găsit o potrivire cu
                          deplasarea", i - m
      k ← suprapunere_k
    sfârșit dacă
  sfârșit pentru
sfârșit algoritm
```

Complexitate

Folosind analiza amortizată putem arăta că algoritmul `Preprocesare` are ordinul de complexitate $O(m)$. Valoarea k poate crește cu cel mult o unitate la fiecare pas al structurii **pentru** și, de fiecare dată când se intră în structura **cât timp**, valoarea k scade. Instrucțiunile executate în interiorul structurii **pentru** au ordinul de complexitate $O(1)$. Datorită faptului că avem m pași, ordinul de complexitate al algoritmului preprocesare este $O(m)$.

Similar se arată că algoritmul `Căutare_KMP` are ordinul de complexitate $O(n)$.

Astfel, ordinul de complexitate al întregului algoritm este $O(n + m)$.

Algoritmul Boyer-Moore

Acest algoritm folosește o combinație între doi algoritmi euristici. Timpul de lucru al acestui algoritm poate fi subliniar. În general, timpul de căutare este cu atât mai redus cu cât lungimea șablonului este mai mare.

În continuare este prezentată descrierea în pseudocod a algoritmului. Menționăm că modul în care se construiesc vectorii *badchar* și *goodsuffix* va fi prezentat tot în cadrul acestui articol.

```
algoritm Boyer-Moore(T, S)
  n ← lungime(T)
  m ← lungime(S)
  Bad_Character(S) // aceste funcții vor fi prezentate
  Good_Suffix(S)   // detaliat
  i ← m
  cât timp i ≤ n execută
    j ← m
    cât timp j > 0 și S_j = T_{i-(m-j)} execută
      j ← j - 1
    sfârșit cât timp
    dacă j = 0 atunci
      scrie "Am găsit o potrivire cu
                              deplasarea", i - m
      i ← i + goodsuffix_i
    altfel
      i ← i + max(j - badchar_{T_{i-(m-j)}},
                              goodsuffix_{j+1})
    sfârșit dacă
  sfârșit cât timp
sfârșit algoritm
```

Se observă că acest algoritm este asemănător cu algoritmul naiv de căutare. Spre deosebire însă de acesta din urmă, în faza de comparare, șablonul este parcurs de la sfârșit la început (utilitatea acestei metode se va vedea în descrierea celor două euristici). De asemenea, în cazul în care găsește o nepotrivire, algoritmul naiv mută șablonul doar cu o poziție la dreapta. În acest caz, fiecare algoritm furnizează o poziție pe care șablonul se poate deplasa la dreapta, știind sigur că până acolo nu mai există nicio potrivire, iar algoritmul *Boyer-Moore* alege maximul dintre aceste două poziții.

În continuare vom descrie cele două euristici care stau la baza acestui algoritm.

Algoritmul "caracter nepotrivit" (engl. "Bad character heuristics")

Pentru fiecare caracter din alfabet încercăm să găsim cea mai din dreapta poziție din șablon pe care se află acesta. Astfel, în momentul în care găsim o nepotrivire mutăm șablonul la dreapta până când un caracter din șablon se potrivește cu cel din text, așa cum se poate observa în următorul exemplu.

```
1 2 3 4 5 6 7 8 9
a b b a b c b a a
a a b c
    a a b c
```

Comparația dintre `'a'` și `'c'` (pe poziția a patra) eșuează. În șablon, caracterul `'a'` apare pe poziția 1 și pe poziția 2. Așadar, putem muta șablonul cu două poziții la dreapta (în așa fel încat `'a'` - ul din șablon să fie aliniat cu `'a'` - ul din text).

Construcția vectorului *badchar* se realizează astfel:

```
algoritm Bad_Character(S)
  m ← lungime(S)
  pentru fiecare literă x a alfabetului execută
    badchar_x ← 0
  sfârșit pentru
```

```
  pentru i ← 1, m execută
    badchar[S_i] ← i
  sfârșit pentru
sfârșit algoritm
```

Algoritmul "sufix potrivit" (engl. "Good-suffix heuristics")

În momentul în care am găsit o neconcordanță între șablon și text, încercăm să deplasăm șablonul la dreapta astfel încât să existe o potrivire între el și caracterele care au coincis până în acel moment.

Să considerăm următorul exemplu:

```
  1  2  3  4  5  6  7  8  9
  a  b  b  b  a  c  b  a  a
  b  b  a  b  a
           b  b  a  b  a
```

Caracterele "ba" coincid, deci putem muta șablonul la dreapta cu două poziții (până la următoarea apariție a lui "ba" în șablon).

Dacă sufixul nu se găsește în șablon, putem deplasa șablonul cu *m* poziții la dreapta, așa cum se poate observa în cazul următor.

```
  1  2  3  4  5  6  7  8  9
  a  b  b  b  a  c  b  a  a
  b  b  c  b  a
                 b  b  c  ...
```

Sufixul "ba" nu mai apare în șablon, deci ne putem muta cu cinci poziții la dreapta.

Mai trebuie să considerăm cazul în care o parte din sufix se află la începutul șirului:

```
  1  2  3  4  5  6  7  8  9
  a  b  a  b  a  c  b  a  a
  b  a  a  b  a
           b  a  a  b  a
```

În acest exemplu, șirul "aba" de la sfârșitul șablonului s-a potrivit cu textul. Acest șir nu mai apare în șablon, dar nu putem muta șablonul cu cinci poziții la dreapta, deoarece șirul "ba" de la începutul șablonului se potrivește cu sfârșitul șirului "aba". Concluzia este că putem deplasa șablonul doar cu trei poziții.

Prezentăm în continuare modul în care se implementează această euristică. Elementul *goodsuffix*$_i$ reprezintă numărul de poziții cu care trebuie să deplasăm șablonul la dreapta, dacă am găsit o nepotrivire pe poziția i - 1.

Construcția acestui vector este asemănătoare cu faza de preprocesare de la algoritmul *KMP*. Diferența constă în faptul că preprocesăm "inversul" șablonului (adică șablonul citit de la sfârșit spre început).

Astfel, *suprapunere*$_i$ va reprezenta cel mai lung sufix al șirului $S[i \,..\, m]$ care este prefix al șirului $S[i \,..\, m]$. Să presupunem că am găsit un indice i astfel încât $S[i \,..\, m]$ este j. Dacă *goodsuffix*$_j$ nu are atribuită nicio valoare, putem să îi atribuim valoarea $j - i$.

Partea a doua a algoritmului se ocupă de cazul în care numai o parte din sufix se află la începutul șirului.

```
algoritm Good_Suffix(S)
  m ← lungime(S)
  pentru i ← 1, m execută
    goodsuffix_i ← 0
  sfârșit pentru
  k ← m + 1
  suprapunere_m ← k
  pentru i ← m - 1, 1 pas -1 execută
    cât timp k ≤ m și S_k-1 ≠ S_i execută
      dacă goodsuffix_k = 0 atunci
        goodsuffix_k ← k - i - 1
    sfârșit dacă
      k ← suprapunere_k
  sfârșit cât timp
    dacă S_k-1 = S_i atunci
      k ← k - 1
  sfârșit dacă
    suprapunere_i ← k
  sfârșit pentru
  // acum ne ocupăm de al doilea caz
  j ← suprapunere[1]
  pentru i ← 1, m + 1 execută
    dacă goodsuffix_i = 0 atunci
      goodsuffix_i ← j - 1
  sfârșit dacă
    dacă i = j atunci
      j ← suprapunere[j]
  sfârșit dacă
  sfârșit pentru
sfârșit algoritm
```

Variante ale algoritmului

Deoarece euristica sufixului potrivit este greu de înțeles și de implementat, există versiuni ale algoritmului în care această euristică este eliminată.

Un exemplu în acest sens este algoritmul *Horspool*, care folosește doar primul tip de euristică. Spre deosebire însă de algoritmul *Boyer-Moore*, care caută în șablon caracterul ce generează nepotrivirea, acesta caută tot timpul ultimul caracter din șablon.

Algoritmul *Sunday* se aseamănă cu algoritmul *Horspool*, doar că se caută în șablon caracterul din text care urmează după ultimul caracter din șablon, așa cum se poate observa din următorul exemplu.

Algoritmul Horspool

```
  1  2  3  4  5  6  7  8  9
  b  a  b  b  c  c  b  a  a
  b  a  a  b
           b  a  a  b
```

Algoritmul Sunday

```
1 2 3 4 5 6 7 8 9
b a b b c c b a a
b a a b
        b a a b
```

Complexitate

În cel mai defavorabil caz algoritmul *Boyer-Moore* are ordinul de complexitate $O(n \cdot m)$ (în cazul în care avem un text format din n `'a'`-uri și un șablon format din m `'a'`-uri). În cazul cel mai favorabil caz sunt necesare n / m comparații (de exemplu, un text format din n `'a'`-uri și un șablon format din m `'b'`-uri). Șansele de a avea un timp de căutare mic cresc proporțional cu lungimea șablonului și cu numărul de caractere ale alfabetului.

Algoritmul Rabin-Karp

Acesta este un algoritm probabilistic care utilizează tehnica tabelelor de dispersie (engl. *hashing*). Fiecărui șir de caractere îi putem asocia printr-o funcție *hash* un număr natural (pe care îl vom numi în continuare ***valoare hash***).

Numărul tuturor șirurilor diferite este foarte mare și astfel mai multe șiruri pot avea asociat același număr. Algoritmul se bazează pe următoarea proprietate a funcțiilor *hash*: dacă două șiruri au valori *hash* diferite, atunci în mod cert ele sunt distincte; dacă au aceeași valoare *hash*, atunci ele pot coincide sau nu. Algoritmul în pseudocod este:

```
algoritm RabinKarp(T, S)
  hashS ← Hash(S[1 .. m])
  pentru i ← 1, n - m + 1 execută
    dacă Hash(T[i .. i + m - 1]) = hashS atunci
      dacă T[i..i+m-1] = S[1..m]
        scrie "Am găsit o potrivire cu
                          deplasarea", i - 1
    sfârșit dacă
  sfârșit dacă
 sfârșit pentru
sfârșit algoritm
```

Se observă că trebuie să apelăm funcția `Hash` de $n - m + 1$ ori. Cum această funcție are ordinul de complexitate $O(m)$ (trebuie să parcurgem cel puțin odată caracterele din T de la poziția i la poziția $i + m - 1$), complexitatea totală va fi $O((n - m + 1) \cdot m)$, la fel ca cea a algoritmului naiv de căutare.

Pentru a rezolva această problemă folosim o funcție (engl. *rolling hash function*) care calculează în timp constant valoarea *hash* a unui șir $T[p + 1 .. p + m]$, cu ajutorul valorii *hash* a șirului $T[p .. p + m - 1]$.

Exemplu

Să presupunem că avem o funcție `hash` care asociază unui șir suma codurilor *ASCII* ale caracterelor sale. Astfel, avem:

$Hash(T[p + 1 .. p + m]) = Hash(T[p .. p + m - 1]) + Cod\_ASCII(T[p + m]) - Cod\_ASCII(T[p])$.

Această funcție atribuie multor șiruri aceeași valoare și algoritmul nu va sări de multe ori pasul de comparare a șirurilor. Prezentăm în continuare o funcție *hash* mai sofisticată (folosind această funcție șansele ca două șiruri să aibă valorile *hash* egale sunt mici).

Considerăm o funcție care asociază unui șir un număr în baza k, unde k este, de obicei, un număr prim. De exemplu, dacă baza este 13, șirul `"abc"` va avea asociată valoarea $97 \cdot 13^2 + 98 \cdot 13^1 + 98 \cdot 13^0$ (codul *ASCII* al caracterului `'a'` este 97, cel al caracterului `'b'` este 98, iar cel al caracterului `'c'` este 99).

Principalul avantaj al acestei funcții este că valoarea atribuită unui șir se poate calcula din cea atribuită șirului anterior printr-un număr constant de operații. Concret, dacă textul nostru este `"abcdefgh"`, valoarea lui `"bcd"` se obține din valoarea lui `"abc"` scăzând ceea ce am adăugat pentru `'a'` (adică $97 \cdot 13^2$), înmulțind totul cu baza (în cazul nostru 13) și adunând codul *ASCII* al lui `'d'`.

Numerele generate de această funcție pot fi destul de mari și există problema de a nu putea fi stocate în memorie ca tipuri de date standard (ex. `int`, `long`). Pentru a rezolva această problemă, efectuăm toate operațiile modulo p, unde p este, de obicei, un număr prim. Astfel funcția *hash* va avea valori cuprinse între 0 și $p - 1$.

Aplicație

Acest algoritm este folosit pentru a verifica dacă cel puțin unul dintre k șabloane $S_1, S_2, \ldots, S_k$ apare în textul T. El poate efectua această căutare în timp $O(n + k)$ în timp ce ceilalți algoritmi de căutare ar avea ordinul de complexitate $O(n \cdot k)$. Pentru a vedea dacă valoarea *hash* a unui șir este egală cu una dintre cele k valori *hash*, putem folosi un filtru *Bloom* sau o altă structură de reprezentare a mulțimilor care poate verifica apartenența unui element la o mulțime în timp constant (mai multe detalii sunt prezentate în [6]).

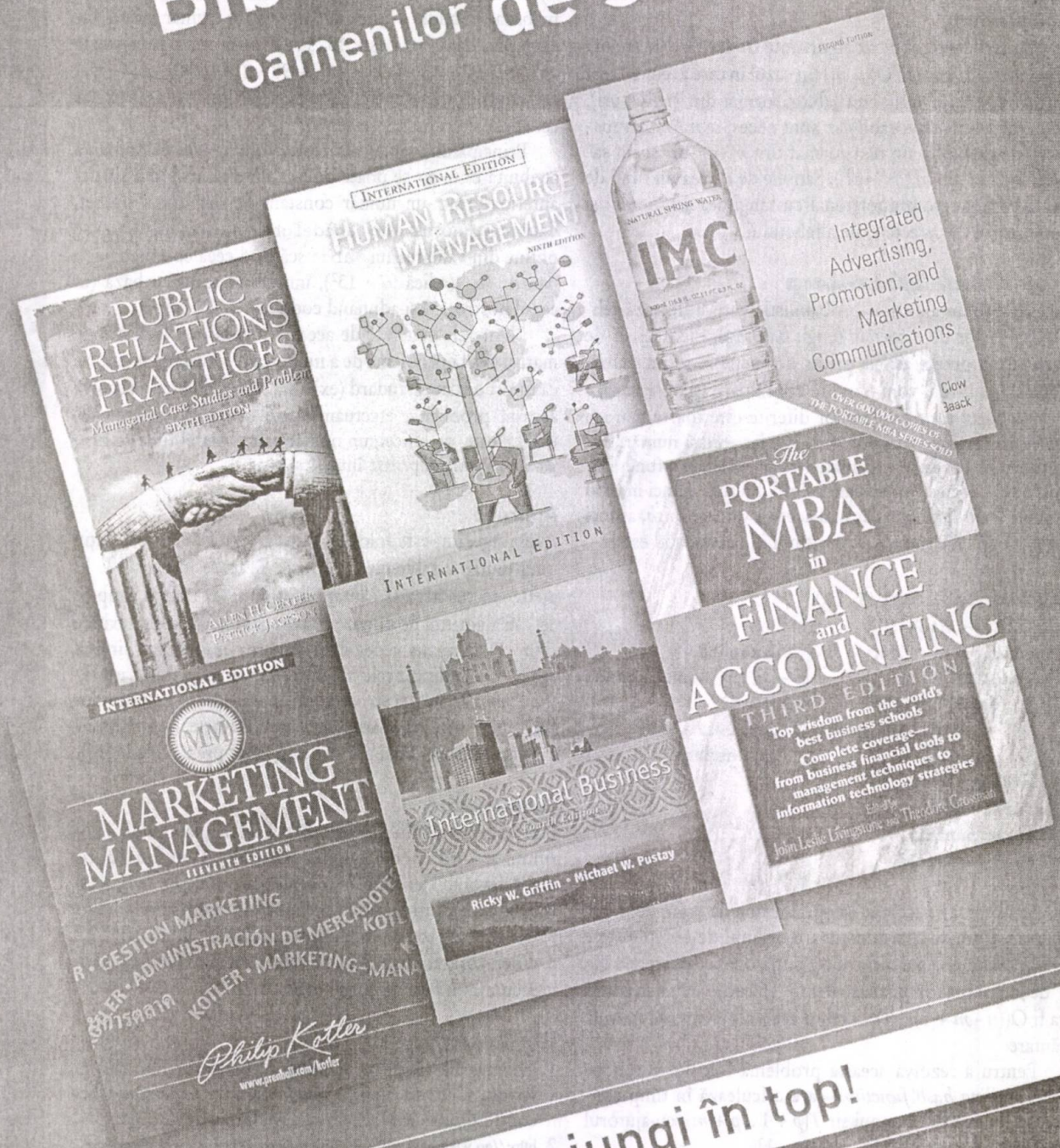
Complexitate

Pentru cazul mediu și pentru cazul cel mai favorabil, algoritmul are ordinul de complexitate $O(n)$. Pentru cazul cel mai defavorabil, ordinul de complexitate este $O(n \cdot m)$, aceasta fiind unul dintre motivele pentru care algoritmul nu este foarte utilizat; totuși, este improbabil să întâlnim cazul cel mai defavorabil (doar dacă folosim o funcție *hash* nepotrivită cum ar fi funcția constantă).

Bibliografie

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, 2001;
2. http://en.wikipedia.org/wiki/Rabin-Karp_string_search_algorithm;
3. http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/bmen.htm;
4. http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/horsen.htm;
5. http://en.wikipedia.org/wiki/Boyer-Moore_string_search_algorithm;
6. http://en.wikipedia.org/wiki/Bloom_filter;
7. http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/sundayen.htm.

Biblioteca
oamenilor de succes



Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

Construirea CUNOAȘTERII

Trecerea de la reproducerea cunoașterii la construirea cunoașterii, utilizând soluții de eLearning

În lumea de azi a devenit redundant să amintești importanța tehnologiilor informației și comunicațiilor. Era informației sau era digitală are un impact major asupra societății în ansamblu și a adus schimbări importante în viața economică și socială.

Existența unui program care să îmbunătățească calitatea și accesibilitatea sistemului educațional prin intermediul integrării efective a tehnologiilor sistemelor informatice este esențială. Procesul de instruire cu ajutorul sistemelor informatice face trecerea de la învățarea bazată pe memorarea mecanică la învățarea bazată pe experiment, de la memorizarea mecanică a informațiilor la construirea cunoașterii.

Introducerea în învățământul românesc a platformei de eLearning AeL, în cadrul programului Sistem Educațional Informatizat al Ministerului Educației și Cercetării, constituie un pas necesar, pe care au început să se construiască bazele unei autentice societăți românești a cunoașterii, competitivă la nivel european.

SEI este o soluție națională de o amploare fără precedent ce are ca obiectiv principal susținerea procesului de predare/învățare în învățământul preuniversitar, cu tehnologii de ultimă oră. Eforturile sunt semnificative și însumează munca unor specialiști din diverse domenii. Similar, rezultatele obținute capătă importanță pe zi ce trece și evaluările ocazionale ale impactului introducerii TIC în școala românească arată dimensiunea schimbărilor majore, de lungă durată, în care sistemul educațional este antrenat.

Dezvoltarea unei viziuni coerente și a unor strategii precise a procesului de informatizare a învățământului vor constitui condiții pentru o reformă bazată pe standarde, ca premisă necesară pentru monitorizarea și evaluarea schimbărilor ce vor avea loc, dar și ca premisă pentru calitatea procesului de educație prin stimularea autoevaluării și posibilitatea recorelării continue cu obiectivele propuse.

Istoria recentă a utilizării calculatorului ca mijloc complementar de educație nu identifică diferențe majore în experiența României față de restul lumii. Perioada anilor 1985 la noi ca și în alte sisteme educaționale din Europa sau din Statele Unite ale Americii consemnează apariția primelor calculatoare dedicate educației, în spațiul școlii.

Primele rețele de calculatoare destinate educației apar la începutul anilor '90. Odată cu prezența INTERNET-ului ca mediu natural de informare, rețelele de calculatoare „didactice" încep să comunice între ele. Apar astfel, primele forme de învățământ colaborativ. Perioada 2006-2010 este descrisă ca fiind de formare a **Comunității Digitale**.

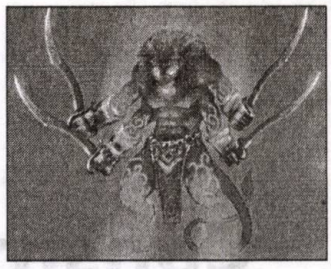
Învățământul bazat pe proiecte devine din ce în ce mai pregnant o formă „obișnuită" de promovare didactică în școli.

Ce ar trebui să urmeze?...spun specialiștii...eLearning! Cuvântul de ordine ar fi **accesul la cunoaștere oricând și oriunde, în timp real.**

Un proiect de eLearning trebuie să aibă în vedere cinci componente fundamentale: Curriculum digital, Dezvoltare profesională, Conectivitate, Tehnologie și Dezvoltarea metodelor de învățare. Fiecare dintre aceste elemente are un caracter obligatoriu pentru definirea unui proiect de succes și evoluția fiecăruia determină fie o etapă istorică fie un proiect nedefinit.

Aceste componente definesc implicit și dimensiunea didactică a unui proiect de eLearning. De la studiul individual către centrarea pe elev, de la reproducerea cunoașterii la crearea cunoașterii, dimensiunile eLearningului în ce privește profitul cognitiv sunt definite de câteva repere. Forma educațională cea mai „săracă" din punct de vedere al profitului cognitiv este definită de componentele: instrucție școlară și testare. Trecerea spre crearea cunoașterii, respectiv centrarea pe elev, se realizează prin intermediul elementelor de simulare, emulărilor de proces și experimentelor virtuale.

Olimpius Istrate,
Radu Jugureanu, Florin Ilia



PARCURGEREA unei structuri ARBORESCENTE de DIRECTOARE

Tiberiu Socaciu

Programul pe care îl vom prezenta mătură[1] un întreg drive, înainte de a începe operația el reținând drive-ul curent precum și directorul curent[2]; evident, după terminarea operațiunii se reface contextul inițial. Partea consistentă a programului este[3] transpunerea apelului recursiv în limbaj de asamblare: trebuie salvate pe stivă contextul local al fiecărui apel al procedurii JOB, context reprezentat de blocurile DTA[4].

Această versiune a programului citește numele *drive*-ului de scanat[5], iar la tratarea unui fișier nu face altceva decât să scrie numele complet al acestuia. Dumneavoastră nu aveți decât de modificat secvența de tratare a unui fișier[6] și, eventual, de adăugat câte ceva la secvența de tratare a unui director[7]. Pentru obținerea fișierului executabil, după editarea sursei, trebuie să efectuați următoarele comenzi[8]:

```
TASM SCAN
TLINK SCAN /T /X
DEL SCAN
```

În continuare prezentăm sursa în asamblare a acestui program.

```
C       SEGMENT
ASSUME CS:C,DS:C,ES:C,SS:C

;ADRESA DTA-ULUI
        ORG     80H
DTA     DB      80H DUP(?)

;AICI INCEPE CODUL
        ORG     100H

;SALT LA PRIMA INSTRUCTIUNE EFFCTIVA A
;PROGRAMULUI
INTRARE:
        JMP     START

;DRIVE-UL DE SCAN-AT (IMPLICIT [/:)
DRIVE   DB      2
;DRIVE-UL CURENT (SE MEMOREAZA PENTRU
;REFACEREA CONTEXTULUI)
CURDRV  DB      ?
;DIRECTORUL CURENT DE PE DRIVE-UL CE VA FI
;SCANAT
CURDIR  DB      '\'
        DB      64 DUP(?)
;UN BUFER NECESAR AFISARI1 COMPLETE A NUMELUI
;DE FISIER
BUFFER  DD      ':\'
        DB      64 DUP(?)
;DIRECTORUL PARINTE
TATA    DB      '..',0
;DIRECTORUL RADACINA
ROOTDIR DB      '\',0
```

[1] *scanează*
[2] *contextul inițial*
[3] *în mod evident*
[4] *analoage în acest caz variabilelor de tip searchrec din Turbo Pascal*
[5] *ca fiind prima literă a primului parametru din linia de comandă*
[6] *care începe la adresa FISIER*
[7] *care începe la adresa DIRECTOR*
[8] *se obține un executabil de 436 de octeți, adică foarte scurt, comparat cu o sursă echivalentă scrisă în alt limbaj*

```
;TOATE FISIERELE
TOATE   DB      '*.*' ,0
;SECVENTA NEWLINE
NL      DB      10,13,'$'

;AICI INCEPE EFECTIV PROGRAMUL
START:
;TOATE OPERATIILE PE SIRURI SE EFECTUEAZA DE
;LA STANGA LA DREAPTA
CLD

;DETERMINA DRIVE-UL DE SCANAT
        MOV     SI,80H
        LODSB
        CMP     AL,0
        JZ      FARA
INCA:
        LODSB
        CMP     AL,' '
        JNG     INCA
        CMP     AL,'A'
        JNGE    MJ
        CMP     AL,'Z'
        JNLE    MJ
        SUB     AL,'a'
MJ:
        SUB     AL,'a'
        MOV     DRIVE,AL
FARA:
;DETERMINA DRIVE-UL CURENT
        MOV     AH,19H
        INT     21H
        MOV     CURDRV,AL

;TRECE PE DRIVE-UL NOU
        MOV     AH,0EH
        MOV     DL,DRIVE
        INT     21H

;DETERMINA DIRECTORUL CURENT
        MOV     AH,47H
        MOV     DL,0
        LEA     SI,CURDIR+1
        INT     21H
        JC      STOP

;TRECE IN RADACINA
        MOV     AH,3BH
        LEA     DX,ROOTDIR
        INT     21H
        JC      STOP

;JOB-UL PROPRIU-ZIS
        CALL    JOB
;REVINE IN DIRECTORUL CURENT
        MOV     AH,3BH
        LEA     DX,CURDIR
        INT     21H

;OPRIREA
STOP:
;TRECE PE DRIVE-UL INITIAL
        MOV     AH,0EH
        MOV     DL,CURDRV
        INT     21H

;SECVENTA CLASICA DE TERMINARE
        MOV     AX,4C00H
        INT     21H

;AICI ARE LOC CAUTAREA DE FISIERE
JOB:
;FINDFIRST = DETERMINA PRIMA INTRARE IN
;DIRECTOR
        MOV AH,4EH
        MOV CX,3FH
        LEA     DX,TOATE
        INT     21H
        JC      GATA

REPEAT:
;FISIER, DIRECTOR SAU ALTCEVA?
        MOV     AH,[DTA+15H]
        AND     AH,18H

;SALT DACA E FISIER
        JZ      FISIER
        MOV     AH,[DTA+15H]
        AND     AH,10H

;SALT DACA E DIRECTOR
JNZ     DIRECTOR

;CONTINUARE ALTFEL (DE EXEMPLU ETICHETA)
;FINDNEXT = DETERMINA URMATOAREA INTRARE IN
;DIRECTOR
FINDNEXT:
        MOV     AH,4FH
        INT     21H
        JNC     REPEAT

GATA:
;MERGE IN DIRECTORUL PARINTE
        MOV     AH,3BH
        LEA     DX,TATA
        INT     21H

;REVENIRE DIN SUBPROGRAM
        RET

;TRATAREA UNUI DIRECTOR
DIRECTOR:
```

```
;DACA E DIRECTORUL . SAU .. SARIM PESTE EL
;NEFIIND CONCLUDENT SCOPURILOR NOASTRE
        CMP     [DTA+1EH],'.'
        JNZ     CONTINUE
        CMp     [DTA+1FH],0
        JZ      FINDNEXT
        CMP     [DTA+1FH],'.'
        JNZ     CONTINUE
        CMP     [DTA+20H],0
        JZ      FINDNEXT
CONTINUE:
;SALVEAZA DTA PE STIVA
        SUB SP,80H
        MOV DI,SP
        LEA SI,DTA
        MOV CX,80H
        REP MOVSB
;CHDIR IN NOUL DIRECTOR
        MOV     AH,3BH
        LEA     DX,[DTA+1EH]
        INT     21H
        JC      PESTE

;APEL RECURSIV PENTRU SUBDIRECTOR DACA S-A
;PUTUT INTRA IN EL (SUNT CAZURI IN CARE NU SE
;POATE INTRA)
        CALL    JOB
PESTE:
;REFACE DTA DE PE STIVA
        LEA     DI,DTA
        MOV     CX,80H
        MOV     SI,SP
        REP     MOVSB
        ADD     SP,80H
        JMP     FINDNEXT

;TRATAREA UNUI FISIER (DEOCAMDATA NU FACEM
;ALTCEVA DECIT SA AFISAM NUMELE COMPLET AL
;FISIERULUI
FISIER:
;SCRIE CALEA COMPLETA (DE LA RADACINA PINA LA
;FISIER

;GETDIR
        MOV     AH,47H
        MOV     DL,0
        LEA     SI,BUFFER+2
        INT     21H
;SCHIMBA PE 0-UL DIN ASCIIZ CU $
        CMP     BUFFER+2,0
        JZ      PTROOT
RP:
        LODSB
        CMP     AL,0
        JNZ     RP
        DEC     SI
        MOV     AL,'\'
        MOV     [SI],AL
        INC     SI
        MOV     AL,'$'
        MOV     [SI],AL
        JMP     PST
PTROOT:
        MOV     [BUFFER+2],'$'
PST:
;SCRIE PE ECRAN NUMELE DRIVE-ULUI
        MOV     DL,DRIVE
        ADD     DL,'A'
        MOV     AH,2
        INT     21H

;SCRIE PE ECRAN CALEA
        MOV     AH,9
        LEA     DX,BUFFER
        INT     21H

;SCRIE NUMFLE FISIERULUI
;SCHIMBA PE ASCIIZ CU $
        LEA     SI,DTA+1EH
        LEA     DI,BUFFER+2
REPETA:
        MOVSB
        DEC     SI
        LODSB
        CMP     AL,0
        JNZ     REPETA
        MOV     AL,'$'
        DEC     DI
        STOSB
;SCRIE PE ECRAN
        LEA     DX,BUFFER+2
        MOV     AH,9
        INT     21H
;TRECE LA LINIE NOUA
        LEA     DX,NL
        MOV     AH,9
        INT     21H

;REVENIREA
        JMP     FINDNEXT
ENDS
END     INTRARE
```

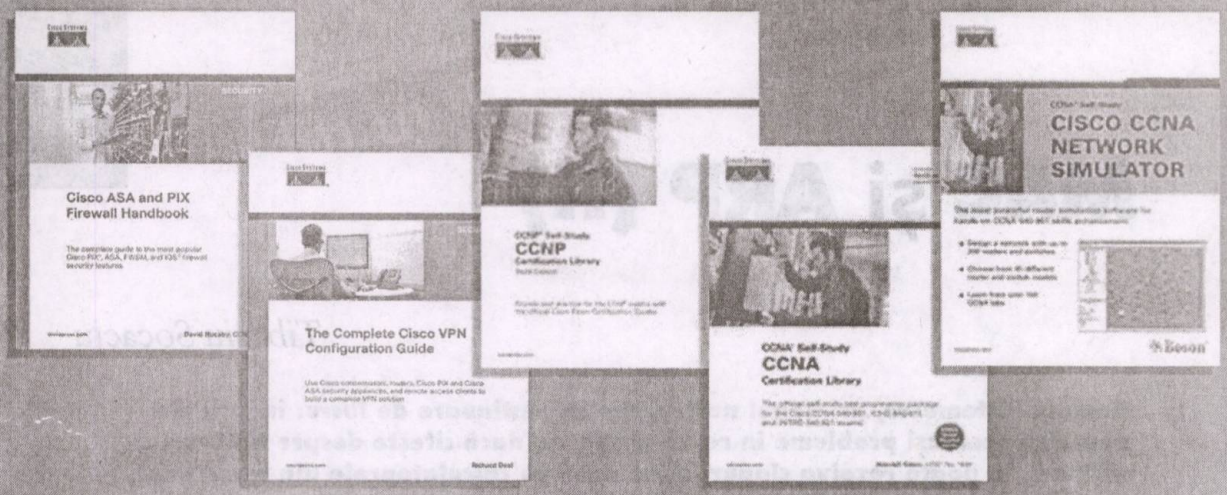
Bibliografie

1. Tiberiu Socaciu, *Comenzi pentru subdirectoare*, în PC Report nr. 9 (iunie 1994), p. 28;
2. Tiberiu Socaciu, *Comenzi pentru grupuri de fișiere*, în if nr. 2/1992 (13), p. 58;
3. Tiberiu Socaciu, *Program cu repetiție*, în *if* nr. 1/1992 (12), p. 53;
4. Tiberiu Socaciu, *Comenzi pentru subdirectoare*, în *if* nr. 1/1992 (12), p. 53.

ISBN: 1587052156
Cisco ASA and PIX Firewall Handbook
Published: Dec 12, 2005

ISBN: 1587201046
CCNP Certification Library
Published: Dec 12, 2003

ISBN: 1587201313
Cisco CCNA Network Simulator
Published: Jul 7, 2004



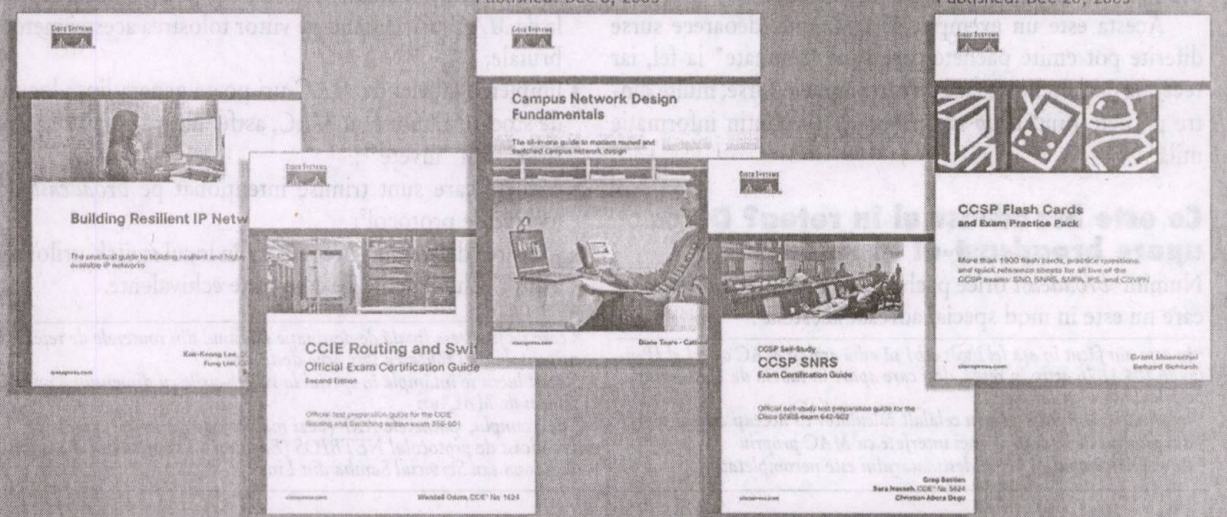
ISBN: 1587052040
The Complete Cisco VPN Configuration Guide
Published: Dec 15, 2005

ISBN: 1587200953
CCNA Certification Library
(CCNA Self-Study, exam #640-801)
Published: Aug 12, 2003

ISBN: 1587052156
Building Resilient IP Networks
Published: Dec 12, 2005

ISBN: 1587052229
Campus Network Design Fundamentals
Published: Dec 8, 2005

ISBN: 158720133
CCSP Flash Cards and Exam Practice Pack
Published: Dec 20, 2005



ISBN: 1587201410
CCIE Routing and Switching Official Exam Certification Guide
Published: Feb 17, 2006

ISBN: 1587201534
CCSP SNRS Exam Certification Guide (Exam Certification Guide)
Published: Dec 8, 2005

Învățătura e o treabă serioasă...

Investește deștept: CISCO PRESS

Cisco Systems
Cisco Press

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

MAC și ARP (II)

Tiberiu Socaciu

Domnul Sistemescu, prietenul nostru, are în continuare de lucru: învață din nou. Are aceleași probleme în rețea și știe că, dacă citește despre MAC-uri și ARP-uri, le poate rezolva singur, chiar dacă pe calculatoarele din rețea sunt, în continuare, activate filtre de tip firewall.

Coliziunea la nivel de MAC

Ce se întâmplă dacă avem două echipamente cu același *MAC* în aceeași rețea? Din cauză că apar două echipamente cu același *MAC*, cele două echipamente nu se vor vedea între ele, dovadă fiind exemplu din casetă[1].

Acesta este un exemplu de coliziune, deoarece surse diferite pot emite pachete care sunt "semnate" la fel, iar recepția pachetelor se face de către ambele surse, multe dintre pachete fiind *drop*-ate deoarece nu conțin informație utilă[2].

Ce este broadcast-ul în rețea? De ce apare broadcast-ul în rețea?

Numim[3] *broadcast* orice pachet care ajunge la o interfață și care nu este în mod special adresat acesteia[4].

Cauzele apariției pachetelor de *broadcast* în rețea pot fi următoarele:

- lipsa informațiilor din tabela de *MAC*-uri a echipamentului; în urma unui *broadcast* "către" o anumită țintă[5], dacă această destinație răspunde, se poate completa tabela de *MAC*-uri, evitând pe viitor folosirea acestei metode brutale;
- umplerea tabelei de *MAC*-uri poate genera lipsa locului de stocare a unui nou *MAC*, astfel încât sistemul va "refuza" să îl "învețe"[6];
- pachete care sunt trimise intenționat pe *broadcast* din motive de protocol[7];
- folosirea de *hub*-uri / repetoare în locul *switch*-urilor sau a altor echipamente de comutare echivalente.

[1] în care am făcut în așa fel încât eth1 să aibă același MAC cu cel al IP-ului 86.104.31.76 activ în retea, deci care apare în tabela de MAC-uri de pe calculatorul cu Linux
[2] informația ar fi utilă pentru celălalt calculator cu aceeași adresă MAC
[3] din punctul de vedere al unei interfețe cu MAC propriu
[4] de regulă câmpul MAC al destinatarului este necompletat
[5] care va fi adresa finală de destinație sau unul din routerele de rețea responsabile cu routarea spre acea destinație
[6] acest lucru se întâmplă în special la switch-urile cu dimensiune mică a tabelei de MAC-uri
[7] de exemplu, pachetele ARP (vezi mai jos) sau pachetele trimise pe broadcast de protocolul NETBIOS (Explorer-ul de network share din Windows sau Serviciul Samba din Linux)

Coliziune MAC

```
[root@server ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
86.104.31.76             ether   00:14:BF:3C:48:FD   C                     eth1
86.104.31.77             ether   00:30:05:29:60:9A   C                     eth1
[root@server ~]# ifconfig eth1 hw ether 00:14:BF:3C:48:FD
[root@server ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
86.104.31.77             ether   00:30:05:29:60:9A   C                     eth1
[root@server ~]# ifconfig eth1 hw ether BE:AC:AF:EA:00:00
[root@server ~]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
86.104.31.76             ether   00:14:BF:3C:48:FD   C                     eth1
86.104.31.77             ether   00:30:05:29:60:9A   C                     eth1
```

Comanda tcpdump (Linux)

```
[root@server ~]# tcpdump --h
tcpdump version 3.9.4
libpcap version 0.9.4
Usage: tcpdump [-aAdDeflLnNOpqRStuUvxX] [-c count] [ -C file_size ]
               [ -E algo:secret ] [ -F file ] [ -i interface ] [ -M secret ]
               [ -r file ] [ -s snaplen ] [ -T type ] [ -w file ]
               [ -W filecount ] [ -y datalinktype ] [ -Z user ]
               [ expression ]

[root@server ~]# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
19:42:57.265764 IP 86.104.31.78.ssh > 86.104.31.77.td-postman: P 1369616948:1369617064(116)
ack 760544215 win 2003
... [ etc etc linii sterse ]
19:44:18.414798 IP 86.104.31.77.td-postman > 86.104.31.78.ssh: . ack 92296 win 61305
19:44:18.414825 IP 86.104.31.77.td-postman > 86.104.31.78.ssh: . ack 92576 win 61270
19:44:18.439556 IP 86.104.31.77.td-postman > 86.104.31.78.ssh: P 11925:11969(44) ack 92660
win 61259
19:44:18.439858 IP 86.104.31.78.ssh > 86.104.31.77.td-postman: P 92660:92760(100) ack 11969
win 2003
19:44:18.439934 IP 86.104.31.78.ssh > 86.104.31.77.td-postman: P 92760:92948(188) ack 11969
win 2003
19:44:18.440116 IP 86.104.31.77.td-postman > 86.104.31.78.ssh: P 11969:12101(132) ack 92760
win 61247
19:44:18.440375 IP 86.104.31.78.ssh > 86.104.31.77.td-postman: P 92948:93216(268) ack 12101
win 2003

770 packets captured
1541 packets received by filter
0 packets dropped by kernel
```

Comanda tcpdump pentru verificarea transmisiilor prin rețea

Pe sisteme *Linux*, pentru a putea verifica pachetele care sunt vehiculate pe interfață putem folosi comanda `tcpdump` a cărei sintaxă este prezentată în casetă (împreună cu un exemplu de verificare a comunicației pe interfața *eth1*[8]).

Pachete ARP

Pachetele *ARP*[9], care mapează adresele *IP* către adresele *hardware* (*MAC*), precum și perechile acestora (pachetele *RARP*[10]) constituie, alături de pachetele *ICMP*, cele mai utilizate protocoale ale *layer*-ului de rețea.

Pachetele *ARP* sunt pachete care sunt vehiculate în rețea. La nivelului protocolului *ARP*, cele mai utilizate tipuri de pachete sunt *ARP WHO-HAS* și *ARP REPLY* care formează perechi. Exemplul din caseta de pe pagina următoare ilustrează modul în care echipamentele sunt obligate[11] să raspundă la un pachet *ARP WHO-HAS* cu un pachet *ARP REPLY*.

Ce se întâmplă la ridicarea interfeței de rețea în Windows?

La ridicarea interfeței de rețea în *Windows* au loc următoarele acțiuni:

- se trimite un *ARP WHO-HAS* pe *broadcast* cu *IP*-ul setat pentru interfața de rețea;
- dacă se primește cel puțin un răspuns *ARP REPLY*, atunci *Windows*-ul refuză să aloce *IP*-ul setat pentru interfața de rețea și va afișa[12] pentru interfața în cauză un mesaj[13] asemănător cu cel din imaginea alăturată.

[8] *am oprit comanda cu CTRL+C; evident puteam să limităm numărul de pachete procesate din linia de comandă*

[9] *Address Resolution Protocol*

[10] *Reverse Address Resolution Protocol*

[11] *indiferent de firewall-ul instalat, care nu are cum opera peste aceste informații*

[12] *pe un sistem cu Windows XP*

[13] *Acquiring network address*

Pachete ARP

```
[root@server ~]# tcpdump -i eth1 -t arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
arp who-has 86.104.31.74 tell 86.104.31.78
arp reply 86.104.31.74 is-at 00:40:f4:b3:8c:96 (oui Unknown)
... [ etc etc linii sterse]
arp who-has 86.104.31.78 (be:ac:af:ea:00:00 (oui Unknown)) tell 86.104.31.76
arp reply 86.104.31.78 is-at be:ac:af:ea:00:00 (oui Unknown)
arp who-has 86.104.31.74 tell 86.104.31.78
arp reply 86.104.31.74 is-at 00:40:f4:b3:8c:96 (oui Unknown)

29 packets captured
58 packets received by filter
0 packets dropped by kernel
```

- dacă nu se primesc răspunsuri *ARP REPLY*, atunci *Windows*-ul alocă *IP*-ul setat pentru interfața de rețea.

Ce se întâmplă în Windows dacă mai apare în rețea încă un calculator cu același IP?

Să simulăm un "accident" prin apariția încă unui *IP* în rețea, prin modificarea *IP*-ului pe interfața *eth0* în mod corespunzător. Astfel, în timp ce pe primul calculator[14] avem *IP*-ul 192.168.71.1 (așa cum se poate vedea în caseta de mai jos), vom încerca să adăugam același *IP* pe cel de-al doilea calculator[15]:

```
# ifconfig eth0 192.168.71.1 netmask
                                255.255.255.0
```

Deoarece un sistem *Windows XP* cu un *IP* dat detectează[16] existența încă unui echipament care are același *IP*, apare o eroare asemănătoare cu cea din imaginea prezentată în continuare:

[14] *Windows XP*

[15] *un sistem Linux*

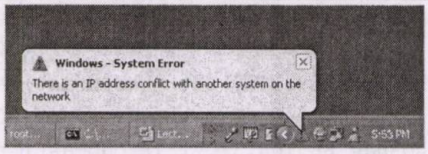
[16] *folosind mecanisme asemănătoare celor prezentate în secțiunea anterioară*

Exemplu (comanda IPCONFIG)

```
C:\Documents and Settings\socaciu.tiberiu>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : internet2
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Hybrid
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Broadcom NetXtreme Gigabit Ethernet for hp
        Physical Address. . . . . . . . . : 00-11-85-62-C1-1C
        Dhcp Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 192.168.71.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.71.254
        DNS Servers . . . . . . . . . . . : 192.168.71.254
                                            193.231.100.2
```

Comanda arping

```
[root@server ~]# arping
Usage: arping [-fqbDUAV] [-c count] [-w timeout] [-I device] [-s source] destination
  -f : quit on first reply
  -q : be quiet
  -b : keep broadcasting, don't go unicast
  -D : duplicate address detection mode
  -U : Unsolicited ARP mode, update your neighbours
  -A : ARP answer mode, update your neighbours
  -V : print version and exit
  -c count : how many packets to send
  -w timeout : how long to wait for a reply
  -I device : which ethernet device to use (eth0)
  -s source : source ip address
  destination : ask for what ip address
```



Vă rugăm să refaceți experimentul cu un sistem din seria *Windows 9x* pentru a vedea eventualele diferențe.

Comanda arping ca alternativă la ping

După cum observați, orice echipament sau calculator răspunde la pachete *ARP WHO-HAS*, chiar dacă are instalate *firewall*-uri, care ar filtra în mod normal pachetele *ICMP*[17]. Prin urmare, programe de tipul `ping` nu sunt întotdeauna utile pentru a verifica dacă un anume *IP* este "*alive*".

Alternativa o constituie programe de tipul `arping`, programe care folosesc secvența de pachete *ARP WHO-HAS* și *ARP-REPLY*. Nu există astfel de programe implementate în mod nativ sub sistemul *Windows*; în schimb, sub *Linux* există posibilitatea de a le folosi. Sintaxa comenzii `arping` este prezentată în caseta[18] de mai sus.

Astfel, pentru a testa dacă *IP*-ul 86.104.31.75 este "*alive*" vom folosi succesiv comenzile `ping` și `arping` cu câte patru pachete trimise[19] pentru a vedea diferențele de output (prezentate în caseta de mai jos).

Studiul execuției unei comenzi arping

Vom folosi două console pe același sistem *Linux*: pe prima consolă vom rula un *dumper* de pachete care va filtra numai pachetele de tip *ARP* legate de *IP*-urile pe care le vom testa, iar în a doua consolă vom da comenzi `arping` către două *IP*-uri (86.104.31.71 și 86.104.31.76). Primul *IP* nu este alocat niciunui echipament activ, în timp ce al doilea

[17] *pachete generate de comenzi de tipul ping*

[18] *comandă rulată pe un sistem cu Fedora Core 5 cu actualizările la zi*

[19] *idem*

Comenzile ping și arping

```
[root@server ~]# ping 86.104.31.75 -c 4
PING 86.104.31.75 (86.104.31.75) 56(84) bytes of data.
64 bytes from 86.104.31.75: icmp_seq=1 ttl=64 time=2.59 ms
64 bytes from 86.104.31.75: icmp_seq=2 ttl=64 time=0.382 ms
64 bytes from 86.104.31.75: icmp_seq=3 ttl=64 time=1.41 ms
64 bytes from 86.104.31.75: icmp_seq=4 ttl=64 time=0.394 ms

--- 86.104.31.75 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.382/1.195/2.591/0.908 ms
[root@server ~]# arping 86.104.31.75 -c 4 -I eth1
ARPING 86.104.31.75 from 86.104.31.78 eth1
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  1.018ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  1.811ms
```

Comanda arping (IP-uri duplicate)

```
[root@server ~]# arping -I eth1 86.104.31.75
ARPING 86.104.31.75 from 86.104.31.78 eth1
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  1.764ms
Unicast reply from 86.104.31.75 [00:40:F4:B3:8C:96]  1.828ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  0.713ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  0.780ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  0.711ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  0.692ms
Unicast reply from 86.104.31.75 [00:06:4F:36:8C:15]  0.741ms
Sent 7 probes (1 broadcast(s))
Received 7 response(s)
```

IP este alocat unui echipament. Astfel, pe prima consolă vom lansa comanda:

```
# tcpdump -i eth1 -n host 86.104.31.75 or
                        86.104.31.76 and -t arp
```

Pe cea de-a doua consolă vom lansa întâi comanda:

```
[root@server ~]# arping -I eth1 -c 4
                                    86.104.31.76
ARPING 86.104.31.76 from 86.104.31.78 eth1
Unicast reply from 86.104.31.76
                    [00:14:BF:3C:48:FD]  1.998ms
Unicast reply from 86.104.31.76
                    [00:14:BF:3C:48:FD]  0.917ms
```

Această comandă generează în prima consolă următorul trafic:

```
[root@server ~]# tcpdump -i eth1 -n host
      86.104.31.71 or 86.104.31.76 and -t arp
tcpdump: verbose output suppressed, use -v or
                  -vv for full protocol decode
listening on eth1, link-type EN10MB
             (Ethernet), capture size 96 bytes
arp who-has 86.104.31.76 (Broadcast) tell
                                    86.104.31.78
arp reply 86.104.31.76 is-at 00:14:bf:3c:48:fd
arp who-has 86.104.31.76 (00:14:bf:3c:48:fd)
                              tell 86.104.31.78
arp reply 86.104.31.76 is-at 00:14:bf:3c:48:fd
```

În continuare, lansăm comanda:

```
[root@server ~]# arping -I eth1 -c 4
                                    86.104.31.71
ARPING 86.104.31.71 from 86.104.31.78 eth1
Sent 4 probes (4 broadcast(s))
Received 0 response(s)
```

Această comandă va genera traficul:

```
[root@server ~]# tcpdump -i eth1 -n host
      86.104.31.71 or 86.104.31.76 and -t arp
tcpdump: verbose output suppressed, use -v or
                  -vv for full protocol decode
listening on eth1, link-type EN10MB
             (Ethernet), capture size 96 bytes
arp who-has 86.104.31.71 (Broadcast) tell
                                    86.104.31.78
arp who-has 86.104.31.71 (Broadcast) tell
                                    86.104.31.78
arp who-has 86.104.31.71 (Broadcast) tell
                                    86.104.31.78
arp who-has 86.104.31.71 (Broadcast) tell
                                    86.104.31.78
```

După cum se observă, nu ne răspunde nimeni la pachetele trimise pe *broadcast*, în timp ce la prima comandă avem succesiunea de pachete (*ARP WHO-HAS*, *ARP REPLY*).

Studiul execuției unei comenzi arping în cazul în care apar mai multe echipamente cu același IP

Să presupunem că avem o rețea în care avem două sau mai multe echipamente cu acelasi *IP*. Cum putem să le identificăm? Simplu: la un pachet *ARP WHO-HAS* sunt obligate să raspundă toate cu *ARP REPLY*. Prin urmare, folosind comanda `arping`, vom putea verifica dacă în rețeaua noastră nu am introdus din "greșeală" încă un echipament cu *IP*-ul 86.104.31.75, pe care l-am alocat unei stații *Windows* 98.

Vom folosi ca și al doilea "cobai" un echipament *Cisco Linksys WRT 54G* care are *firmware*-ul rescris cu *Linux OpenWRT* și un sistem *Fedora Core 5* ca și sistem de monitorizare.

Pentru început, pe sistemul *Linksys* vom seta *IP*-ul ca pentru un echipament deja existent în rețea:

```
# ifconfig vlan1 86.104.31.75 netmask
                                255.255.255.0
```

Acum vom merge pe sistemul *Linux* de monitorizare și vom da comanda `arping` pentru *IP*-ul nostru. Efectul este prezentat în casetă.

Dupa cum se observă, am detectat cele două *MAC*-uri ale echipamentelor care au același *IP*. Dar, ulterior, ne răspunde doar echipamentul aflat la distanța cea mai mică (în timp).

ARP flood

De multe ori, într-o rețea "aglomerată", apar multe pachete *broadcast*, de regulă, pachete tip *ARP* sau pachete uzuale, dar care nu au fost procesate în mod corespunzător de *switch*-uri. De ce apar ele? Explicațiile sunt de doua tipuri:

- pachete trimise în mod explicit pe *broadcast*;
- pachete care sunt procesate în acest mod de către *switch*-uri (de regulă, din motive imputabile umplerii tabelei de *MAC*-uri);

Indiferent care este cauza care le generează, într-o rețea efectul acestui *flood* este foarte neplăcut, scăzând parametri de funcționare în mod vizibil.

Probleme

Deoarece am primit sugestia ca probleme să apară în fiecare număr, chiar dacă materialul are mai multe părți, ne-am conformat: mai jos aveți problemele pentru primele două părți ale articolului despre *MAC* și *ARP*. De asemenea, ni s-a sugerat ca rezolvările să apară în același număr, așa că ne-am conformat. Menționăm că în numărul următor lecția de rețele va fi *MAC și ARP (partea a treia): Protecția la nivel de MAC și IP*.

Problema #10

Sunteți în *Windows* și nu aveți acces la o aplicație de tip `arping` pentru a verifica dacă un *IP* este sau nu "*alive*". Cum procedați pentru a afla, totuși, răspunsul?

Rezolvare

Dați un `ping` către respectivul *IP*. Această comandă generează un pachet *ARP WHO-HAS*, la care se primește obligatoriu răspuns, chiar dacă pachetele *ICMP* sunt filtrate de către *host*-ul în cauză. Imediat după comanda `ping`, dați comanda `arp` pentru a inspecta tabela de *MAC*-uri, de unde veți obține informația dacă *host*-ul este "*alive*" sau nu.

Problema #11

Sunteți legat într-un *switch* și rulați o aplicație de tip `tcpdump`. În *switch* nu mai este conectat nimeni, în afară de legătura spre restul rețelei. Vedeți comunicație intensă, care nu este de tip *broadcast*, între alți doi membri ai rețelei. Cum vă explicați?

Rezolvare

Din diverse motive (de regulă depășirea tabelelor de *MAC* ale *switch*-urilor de pe traseu) unele *switch*-uri se comportă în trafic ca niște *hub*-uri.

Problema #12

De ce nu este bine să se conecteze în rețele de cartier echipamente *wireless* setate în mod *bridge*?

Rezolvare

Rețelele de cartier sunt binecunoscute ca fiind generatoare de trafic *broadcast* (în special pachete *ARP*) imens. Datorită intensității acestui trafic rezidual, o comunicare *WiFi* nu rezistă.

Problema #13

Aveți un echipament de tip *cable-modem* (sau *ADSL-modem*). Presupunem că echipamentul poate fi setat ca *bridge* (*modem only*) sau ca *router* (*router-modem*). Cum este mai bine să setăm acest echipament?

Rezolvare

Dacă aveți de conectat un singur echipament (care nu este un *WiFi* în *bridge*) atunci, eventual, echipamentul *cable-modem* poate fi setat ca *bridge*.

Dacă aveți de conectat mai multe echipamente, atunci va trebui setat ca *router*.

Problema #14

De ce anumite *switch*-uri se comportă în trafic, practic, ca niște *hub*-uri?

Rezolvare

De regulă aceste *switch*-uri au tabela de *MAC*-uri prea mică, astfel încât multe dintre pachete nu pot fi comutate și se retransmit pe toate porturile.

Problema #15

Aveti două rețele interconectate via un tunel *TAP*, tunelul *TAP* fiind pus în *bridge* cu cele două rețele locale via *router*-ele *Linux* care le deservesc. Pachetele *ARP* emise de calculatoarele din rețeaua 1 ajung să fie pe *broadcast* și în rețeaua 2?

Este o soluție viabilă din punctul de vedere al unei rețele de cartier?

Rezolvare

Este evident că răspunsul este ***da***. Soluția este viabilă în momentul în care *bandwidth*-ul alocat suportului de tunel este cel puțin comparabil cu *bandwidth*-ul de *broadcast* însumat din cele două rețele.

Problema #16

Sunteți un *ISP* care operează pe o mare rețea de cartier. Observați că aveți foarte multe pachete *broadcast*, din diverse motive. Ce soluție adoptați?

Rezolvare

Decizia corectă este de a "rupe" rețeaua introducând *router*-e de acces, de exemplu sisteme cu *Linux*. Un argument suplimentar este posibilitatea de a genera și redundanță dacă se trece la un nivel superior de rutare între ruterele de acces (de exemplu *BGP*).

Problema #17

Care este diferența dintre un mediu "*shared*" (de exemplu, *hub*-ul) și un mediu "*switched*" (de exemplu *switch*-ul) în comunicația *Ethernet*?

Rezolvare

Într-un mediu *shared* orice pachet recepționat pe un anumit port este retransmis pe toate celelalte porturi; într-un mediu *switched*, pachetele se retransmit, pe cât posibil, doar pe un singur port.

Problema #18

De ce, în trafic, unele *switch*-uri se blochează și după un simplu reset, ele funcționează normal?

Rezolvare

Singura explicație *software* (trecând peste cele de tip *hardware*) este blocarea *software*-ului în momentul umplerii tabelei de *MAC*-uri sau la imposibilitatea comutării unor pachete după umplerea tabelei.

Problema #19

Aveți o rețea de cartier în care foarte mult trafic poate fi catalogat ca fiind "arp flood". Ce măsuri luați?

Rezolvare

Soluția constă în "ruperea" rețelei și introducerea a cât mai multor *router*-e de acces locale.

Problema #20

De ce, în trafic, unele *switch*-uri se blochează și după timp de câteva secunde, ele funcționează normal?

Rezolvare

Singura explicație *software* (trecând peste cele de tip *hardware*) este umplerea tabelei de *MAC*-uri, după care urmează o golire a acestei tabele sau o resetare a echipamentului.

Bibliografie

1. **Tiberiu Socaciu**, *Lecții de rețele*, vol. 1, Editura InfoData, Cluj-Napoca, 2007;
2. **Tiberiu Socaciu**, *MAC și ARP (I)*, *GInfo* 17/2, februarie 2007, Editura Agora Media, Târgu Mureș;
3. **Tiberiu Socaciu**, *VLAN, TRUNK și STP*, *GInfo* 17/1, ianuarie 2007, Editura Agora Media, Târgu Mureș;
4. **Tiberiu Socaciu**, *Tuneluri și bridge-uri*, *GInfo* 16/8, decembrie 2006, Editura Agora Media, Târgu Mureș;
5. **Tiberiu Socaciu**, *Limitări de viteză în rețele*, *GInfo* 16/6, octombrie 2006, Editura Agora Media, Târgu Mureș;
6. **Tiberiu Socaciu**, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft, Bacău, 2006;
7. **Tiberiu Socaciu**, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura InfoData, Cluj-Napoca, 2006;
8. **Tiberiu Socaciu**, *O altfel de ... Informatică sau diverse chestiuni de matematică, backtracking, clustere de calculatoare, Linux, inteligență artificială și networking*, Editura InfoData, Cluj-Napoca, 2007.

HDD de 1 TB

La începutul anului compania **Hitachi** a anunțat că intenționează să lanseze pe piață primul disc hard cu capacitatea de stocare de un terabyte.

Vor exista, pentru început, două variante ale produsului: **Deskstar 7K1000** și **Cinemastar 7K1000**. Prima variantă este destinată celor care doresc să stocheze, să păstreze sau să distribuie informație digitală, în timp ce cea de-a doua variantă promite optimizări destinate în special celor care realizează înregistrări video.

Noile discuri au o viteză de rotație de 7200 rpm, folosesc interfața SATA care permite transferuri la viteze de până la 3 GB/s și vor avea, la lansare, prețuri începând cu 399$.

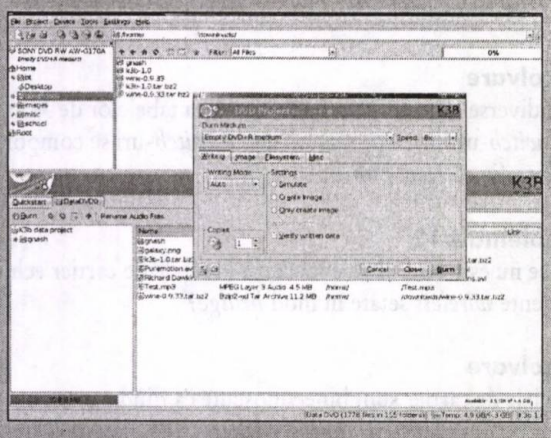
Atât pentru modelul **Deskstar**, cât și pentru **Cinemastar**, vor exista și variante cu capacitatea de stocare de 750 GB.

Lansarea noilor discuri este preconizată a avea loc în prima jumătate a acestui an.

K3B 1.0

După aproape zece ani de dezvoltare, timp în care și-a câștigat renumele de cel mai bun utilitar pentru inscripționarea discurilor optice sub Linux, **K3B** a ajuns la versiunea finală pe data de 16 martie 2007.

Practic **K3B** folosește comenzile Linux cdrecord, cdrdao, and growisofs, dar oferă o interfață grafică plăcută, care rivalizează cu orice altă aplicație concurentă.



Mașini inteligente (1)

INTELIGENȚA ARTIFICIALĂ între ADEVĂR și MIT

Ionuț Chirilă

Aproape toată lumea a auzit cel puțin o dată despre conceptul de inteligență artificială. Când vine vorba de aplicarea ei sau chiar o definire sumară, părerile sunt deja împărțite. Ideea centrală este foarte simplă: cum putem transforma calculatorul dintr-un executant mai mult sau mai puțin docil într-un adevărat partener? Fără riscul de a plictisi cu date seci despre istoricul acestui concept și primele sale interpretări, vom începe să sintetizăm ideile importante pe care se sprijină implementarea în practică.

Un calculator execută un algoritm în programarea clasică. În *IA*, el trebuie să fie învățat să ia decizii, pe baza unor reguli de producție concepute de inginerul uman.

Există similitudini evidente cu modelul de organizare al creierului nostru cu principala deosebire că acesta modelează cu viteză redusă față de procesorul electronic al calculatorului personal.

Regulile de producție mai sunt numite în sfera *IA* și *reguli de inferență*, pe baza cărora este construit motorul de căutare al informației necesare problemei de rezolvat.

Este o chestiune de timp până când sistemele expert bazate pe inteligența artificială vor fi nelipsite din mai toate domeniile de activitate: medicina, comerț etc.

Ele nu vor înlocui în totalitate factorul uman, dar vor contribui din plin la luarea deciziilor. Aria de expertiză a acestor sisteme cuprinde la ora actuală probleme de selecție, diagnoză, sinteză.

Să ne închipuim cum ar ușura munca umană și timpul consumat un sistem expert de diagnoză a unei mașini sau ce ar însemna pentru un medic posibilitatea consultarii permanente cu *PC*-ul în ceea ce privește medicamentația pacienților săi.

Alte importante aplicări ale inteligenței artificiale le constituie recunoașterea vocii, jocurile sau chiar perceperea limbajului natural. Un mare avantaj al unui astfel de sistem este imposibilitatea de a greși de două ori în același loc. El asimilează regulile de producție, construiește modelul (modulul) de căutare a soluției, apoi procesează baza de cunoștinte inițială pentru găsirea soluțiilor optime. Ca și în programarea clasică, și aici, în anumite cazuri, se poate căuta soluția optimă între optimele parțiale obținute.

În cazul problemelor de diagnoză și clasificare/selecție se pornește de la mulțimea de fapte inițiale, nemodelate.

Ca un exemplu posibil, să considerăm că un restaurant franțuzesc dorește să "potrivească" felurile de brânză din meniu cu tipurile de vinuri disponibile.

Înseamnă că în baza de cunoștințe vom avea, pe de o parte, fiecare vin cu particularitățile sale, iar pe de altă parte, felurile de brânză, în același mod, cu câteva caracteristici ale acestora.

Modelul selectat pentru interogarea bazei de cunoștințe în vederea găsirii soluțiilor va fi cel al interviului. Pentru construirea regulilor de producție vom căuta similitudini cu lumea reală.

De exemplu, o regulă de producție ar putea fi: dacă brânza este dulce, atunci vinul cel mai potrivit este *Cabernet* sec cu o probabilitate de 75% și vinul cel mai potrivit este *Merlot* cu o probabilitate de 50% etc.

Un aspect foarte important îl constituie stabilirea acestor indici ai probabilității, numiți în limbajele de inteligență artificială *factori de încredere*.

Fiecărei reguli i se poate asocia un grad de încredere (așa-numitul *salience*) între 0 și 100. Combinarea factorilor de încredere se realizează la alegere, în funcție de situație, prin metode precum: metoda produselor, metoda maximum sau minimum.

Numărul regulilor de producție, precum și respectarea unui model real cât mai corect va duce la obținerea unor rezultate mai fine. În cele mai multe situații soluția obținută nu este unică.

De asemenea, este nevoie de o bază de cunoștințe atât de bogată încât să obțină soluții indiferent de răspunsurile oferite sistemului de către utilizatorul uman.

Interogarea are rolul ei, atât la construcția arborelui de căutare, cât și prin natura variantelor de răspuns lăsate pentru alegere.

Arborele de căutare poate fi cu înlănțuire înainte sau înlănțuire înapoi. În practică, se întâlnesc și situații în care cele două moduri de cautare apar îmbinate.

Unele oferă o viteză mărită de căutare, dar o anumită *superficialitate* deoarece înlănțuirea înapoi determină care reguli din mulțimea inițială vor fi selectate și folosite, în timp ce înlănțuirea înainte pornește de la datele deja existente urmând, prin folosirea motorului de inferență, să extragă mai multe date.

În final, soluțiile obținute pot fi, la prima vedere, comparate cu ochiul liber, urmărind în ce măsură respectă atributele inițiale ale faptelor.

O soluție care va respecta mai multe atribute va avea, fără îndoială, o finețe ridicată.

Menționăm faptul că o aplicare a ambelor modalități pentru aceeași problemă poate duce la soluții diferite. Acest lucru subliniază încă odată faptul că trebuie respectat cât de mult posibil modelul real al problemei, și anume care dintre cele două tipuri de înlănțuiri se poate aplica logic în situația dată.

Am arătat, punctând lucrurile esențiale, cum funcționează un sistem bazat pe inteligența artificială, în cazul unei probleme de clasificare/selectie.

Rămânând în jurul ideii centrale a titlului acestui articol, putem afirma că, folosind pași mărunți (inductivi sau deductivi), putem afla în timp despre *IA* aproape la fel de mult cât cunoaște *Bill Gates* despre sistemul de operare *Microsoft Windows*.

IA în ficțiune

În science fiction inteligența artificială este prezentată sub forma unei puteri care va răsturna autoritatea umană sau sub forma unor umanoizi care preiau sarcinile ființelor umane.

Din prima categorie fac parte calculatorul **HAL 9000** din **Odiseea spațială 2001** a lui **Arthur C. Clarke** sau rețeaua **Skynet** din seria de filme **Terminator**.

Exemple reprezentative pentru cea de-a doua categorie ar putea fi **C-3PO** din universul **Star Wars** sau **Lt. cmdr. Data** din universul **Star Trek**.

Frank Herbert a sugerat ideea că, la un moment dat, umanitatea ar putea să interzică utilizarea mașinilor inteligente datorită pericolelor cărora ar trebui să le facă față umanitatea. În seria **Dune** este menționată o rebeliune numită **Jihadul Butlerian**, în urma căreia mașinile inteligente sunt învinse și este impusă pedeapsa capitală pentru oricine ar putea încerca să dezvolte astfel de mașini în viitor. Este deseori menționat un citat din **Biblia Catolică Portocalie** în care se spune: **"Thou shalt not make a machine in the likeness of a human mind"**. O abordare similară apare în universul **Battlestar Galactica**.

Legile roboticii

Isaac Asimov a imaginat pentru roboții pe care i-a creat în romanele sale trei legi fundamentale care să modeleze comportamentul acestora:

Legea 1: Un robot nu are voie să pricinuiască vreun rău unei ființe umane sau, prin neintervenție, să permită ca unei ființe omenești să i se facă rău.

Legea 2: Un robot trebuie să se supună ordinelor date de către o ființă umană atât timp cât ele nu intră în contradicție cu Legea 1.

Legea 3: Un robot trebuie să-și protejeze propria existență, atât timp cât acest lucru nu intră în contradicție cu Legea 1 sau Legea 2.

Ulterior, **Asimov** a adăugat o lege suplimentară, care a devenit legea supremă:

Legea 0: Un robot nu are voie să pricinuiască vreun rău umanității sau, prin neintervenție, să permită ca umanitatea să fie pusă în pericol.

Celelalte trei legi sunt modificate în mod corespunzător pentru a nu apărea nicio contradicție cu Legea 0.

Portalul **Agora.ro**

Am concentrat pentru tine Informația.
Navigare ușoară, lectură ușoară.
Poți să știi...

Agora Media
www.agora.ro

Cabinetul
Călin Tatomir

Noi ştim cum

Aprilie 2007
Vol. 17/4

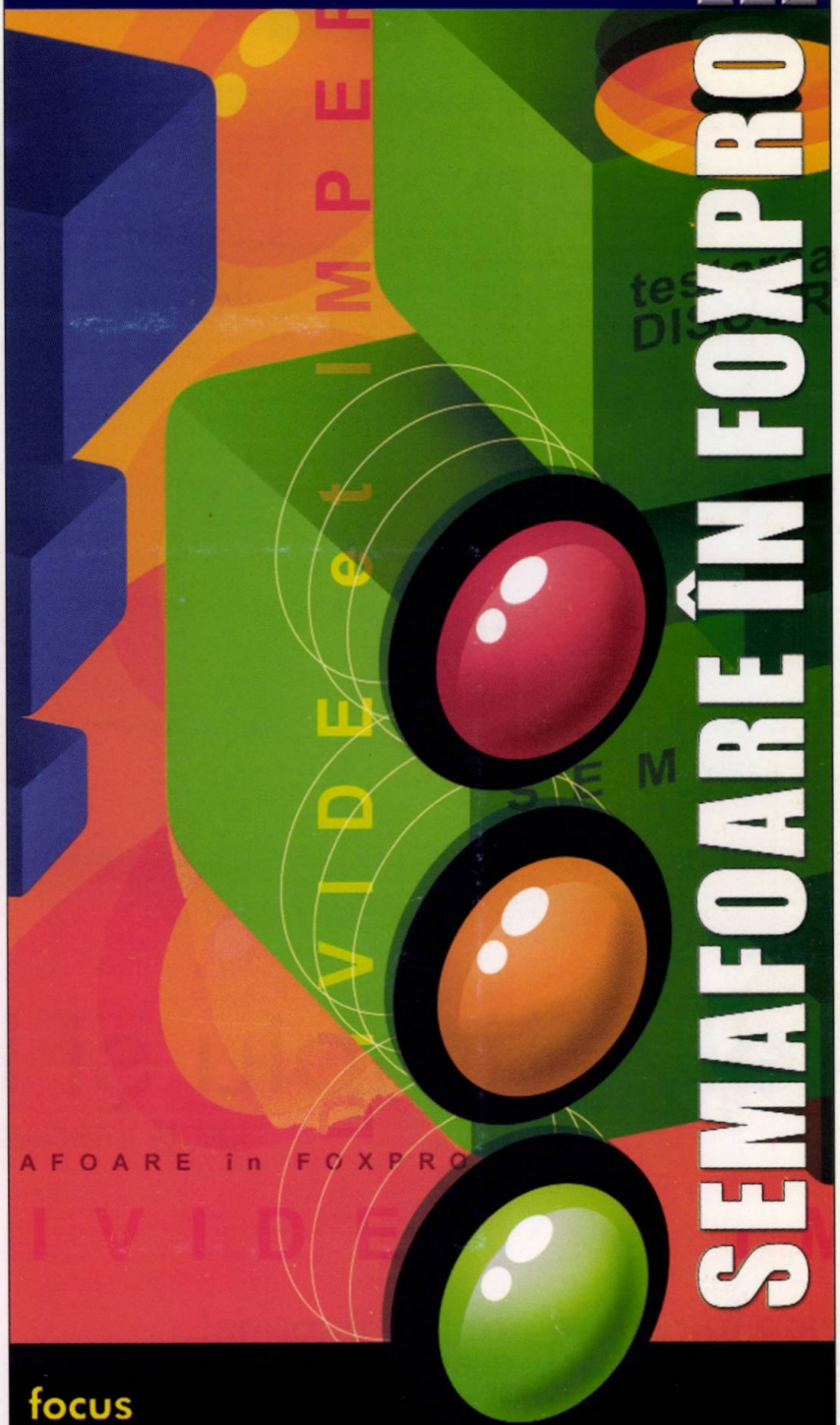
40.000 LEI / 4 LEI NOI

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

o publicație Agora MEDIA

din sumarul numărului:



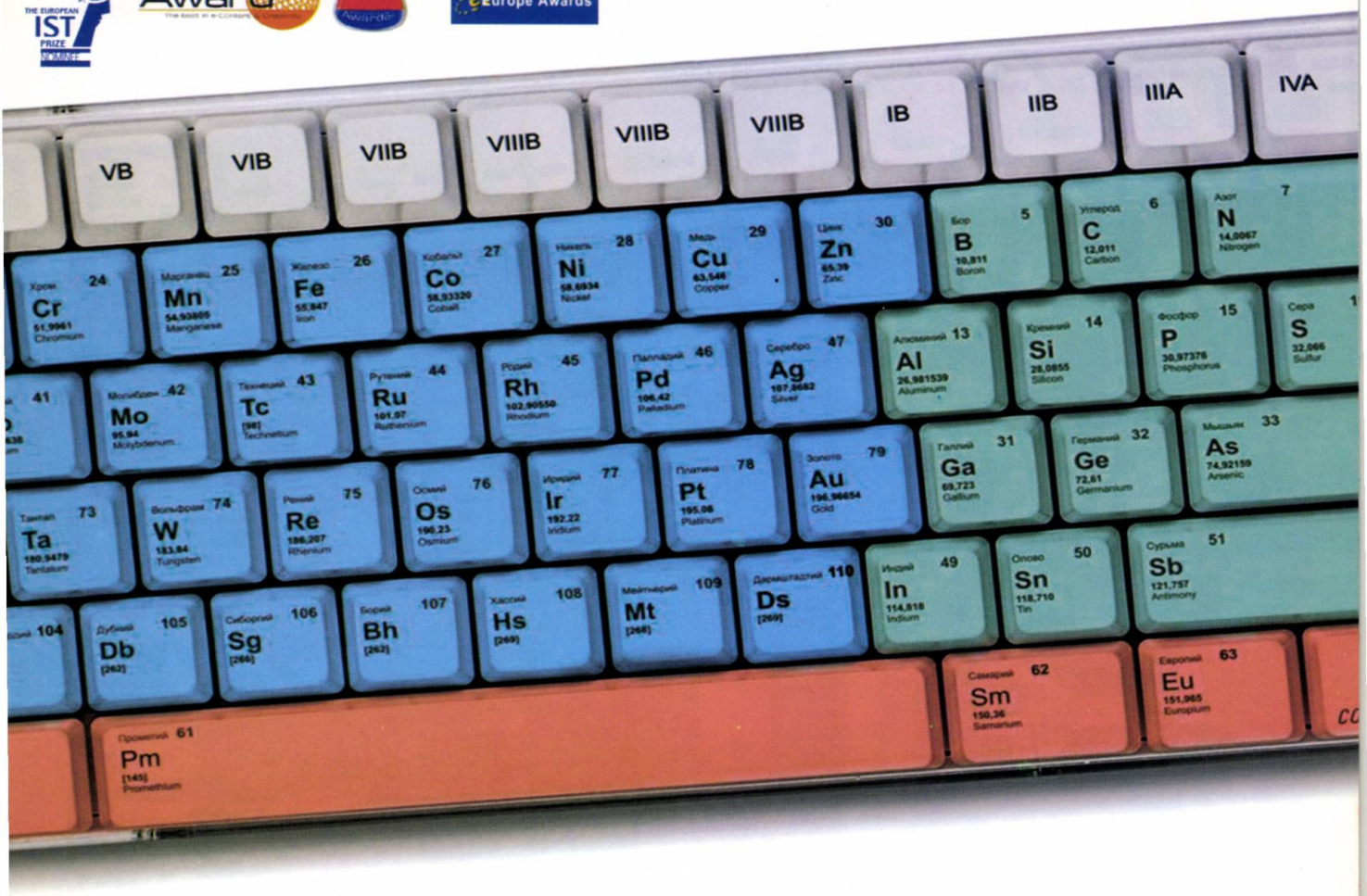
focus
Metoda DIVIDE et IMPERA
Backtracking recursiv

babel
Testarea în FOXPRO

internet
Algoritmul lui DIJKSTRA

enjoy
HEROES of MIGHT and MAGIC

5948373000645 04



AeL. Simplu şi la obiect.

Lecţii interactive pentru noua generaţie.

Orice întrebare are un răspuns!

Noutăţi, suport tehnic, competiţii,
forum : www.portal.edu.ro

Orice problemă are o soluţie!

Centrul de suport tehnic de luni până
vineri între orele 8.00-20.00
Telverde : 0800410444 (linie gratuită)

Orice cauţi poate fi găsit!

Ai nevoie de o anumită lecţie electronică?
Întâi de toate, caut-o în biblioteca virtuală
AeL. Odată găsită, descarc-o gratuit!

AeL
Built to teach intelli-gently

sei

SIVECO
ROMANIA SA

Vol. 17, Nr. 4 /2007

gazeta de informatică

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tiberiu Socaciu
Colaboratori:
Vlad Bazon
Ionuț Chirilă
Ion Petre

Adresa pentru corespondență:
Computer Libris Agora,
Șerpuitoare 25, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/534.911, 0265/266.516
Fax: 0265/266.290

http://www.ginfo.ro
ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL

Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NET Report*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

O publicație Agora MEDIA

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu

Difuzare:
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

focus.. PAG. 5

Metoda DIVIDE et IMPERA
Ion PETRE

PARALELISM și CONCURENȚĂ în algoritmul BACKTRACKING RECURSIV
Tiberiu SOCACIU

babel.. PAG. 14

Testarea DISCURILOR și DIRECTOARELOR în FOXPRO
Tiberiu SOCACIU

Programare CONCURENTĂ și SEMAFOARE în FOXPRO
Tiberiu SOCACIU

internet.. PAG. 21

Model JS-PROTOTYPE pentru ALGORITMUL lui DIJKSTRA
Vlad BAZON

e-learning.. PAG. 27

Construirea CUNOAȘTERII (II)
Radu Jugureanu, Florin Ilia

tehno.. PAG. 28

DIRERASE sau cum să ștergem REPEDE
Tiberiu SOCACIU

serial.. PAG. 30

MAC și ARP (III)
Tiberiu SOCACIU

INTELIGENȚA ARTIFICIALĂ între ADEVĂR și MIT
Ionuț CHIRILĂ

enjoy.. PAG. 36

HEROES of MIGHT and MAGIC V: HAMMERS of FATE
Mihai SCORȚARU

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

Singura condiție pe care trebuie să o îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre.

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii. Pentru mai multe detalii vizitați:
http://www.ginfo.ro/revista/publicare.shtml

Cea mai bună introducere din lume în rețelele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine

EDIȚIA A PATRA

Rețele de calculatoare

ANDREW S. TANENBAUM

Cea mai bună introducere în arhitectura calculatoarelor - complet actualizată, reflectă ultimele schimbări din dinamica lume a calculatoarelor

Andrew S. Tanenbaum

ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR

Ediția a IV-a

SISTEME DE OPERARE MODERNE

EDIȚIA A DOUA

Andrew S. Tanenbaum

THOMAS H. CORMEN · CHARLES E. LEISERSON · RONALD R. RIVEST

INTRODUCERE ÎN ALGORITMI

Această carte de referință, bazată pe exemple concrete, oferă informații practice de primă mână pentru înțelegerea și construirea sistemelor de operare moderne.

Cartea tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, fiind binevenita elevilor, studenților și profesorilor

CĂRȚILE ESENȚIALE ▸

...cărțile care îți însoțesc cariera

Comandă-le pe toate și le iei cu 20% mai ieftin.
Iar dacă ești student, cu 30%.

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

ERORI interesante

Există o mulțime de cărți în care apar erori ciudate. Există și o mulțime de ziare și reviste în care apar erori și mai ciudate... GInfo nu face excepție...

Când nu este vorba de *GInfo*, observ relativ ușor greșelile care apar în diverse publicații. Una dintre cele mai interesante "gafe" am văzut-o într-o revistă care lăuda prestația unei anumite prezentatoare de știri și, în sprijinul acestor afirmații, apăreau câteva imagini în care se vedea sigla unei alte emisiuni de știri de la o televiziune concurentă.

Erorile de tehnoredactare sunt frecvente (sper că *GInfo* este sub media națională din acest punct de vedere), iar greșelile de exprimare apar la tot pasul (pentru *GInfo* îmi place să cred că nu este cazul).

Totuși, am observat recent o greșeală foarte interesantă, datorată mai multor factori. Citeam o carte foarte bună (în opinia criticilor din întreaga lume), dar a cărei traducere lăsa de dorit. Așadar, lectura era cam greoaie și cartea părea cam plictisitoare...

Totuși, la un moment dat am văzut una dintre erorile care "sar în ochi". Se spunea că, pe baza unor anumite ipoteze, capacitatea creierului de a stoca informația era de o mie douăzeci de biți (era scris așa, cu litere). Cam puțin mi-am zis...

În continuarea raționamentului, se spunea că, dacă luăm în considerare alte ipoteze, atunci capacitatea ar crește foarte mult, până la o mie treizeci și doi de biți (scris tot cu litere). Deja părea interesant; aproximativ 1% înseamnă foarte mult...

Totuși, raționamentul continua și, dacă erau luate în considerare alte ipoteze, atunci capacitatea creștea de câteva zeci de miliarde de ori, până la o mie patruzeci și trei de biți.

Așadar, probabil, autorul a scris 10^{20}, 10^{32} și 10^{43}. Cumva, în timpul conversiei documentului dintr-un format în altul, exponenții au dispărut și numerele au devenit 1020, 1032 și 1043. Dar, traducătorul a avut o idee genială: e mai interesant dacă numerele astea apar scrise cu litere.

Așadar, am observat un efect interesant care a putut apărea doar datorită neatenției mai multor persoane (o traducere este și ea, cel puțin în principiu, redactată și corectată).

Cam atât despre erori... Dacă vedeți astfel de ciudățenii în revista noastră, vă rog să ni le semnalați (ne vom distra împreună). Poate ar trebui să acordăm și premii cititorilor care raportează astfel de erori... Și poate premiile ar trebui scăzute din salariile celor care comit erorile... Și poate așa, toată redacția va ajunge să lucreze cu adevărat "pe gratis"...

Până atunci vă las în compania dwarfilor care vor indica nivelurile de dificultate ale articolelor. Dacă se confirmă anumite ipoteze, în numărul următor vom avea orci. Dacă se confirmă alte ipoteze, vom avea altceva...

Mihai Scorțaru

Cine are carte, are parte!

...de înțelegerea rețelelor!

Cea mai bună introducere din lume în rețelele de calculatoare

- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o REDUCERE semnificativă!

Byblos srl, tel: 0318-056-314, 0721-927-499, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librărie: Galeria NOI-La Calu' Bălan, Str. Halelor Nr. 3 (după Hanul lui Manuc).

Împarte și stăpânește!

Metoda DIVIDE et IMPERA

Ion Petre

În cadrul acestui articol vom prezenta câteva probleme care pot fi rezolvate folosind metoda divide et impera.

Problema tăieturilor în plan

Se consideră o suprafață în formă dreptunghiulară cu laturile *Lx* și *Ly* (exprimate în metri). Laturile dreptunghiului sunt paralele cu un sistem de coordonate cartezian *XOY*, lungimea (*Lx*) fiind paralelă cu axa *OX*, iar lățimea (*Ly*) fiind paralelă cu axa *OY*. Se dau coordonatele (*U*, *V*) ale colțului din stânga-jos (exprimate în metri).

În dreptunghi există *n* puncte. Punctele sunt definite prin coordonatele (X_i, Y_i), unde *i* variază între 1 și *n*.

Să se determine un dreptunghi cu suprafața maximă care să nu conțină puncte în interiorul său. Laturile dreptunghiului căutat trebuie să fie paralele cu laturile dreptunghiului dat inițial. Orice latură a dreptunghiului căutat poate conține puncte și poate fi plasată pe o latură a dreptunghiului dat.

Date de intrare

Datele din fișierul de intrare **TAIEPLAN.IN** sunt structurate astfel:

- pe prima linie se află valorile *n*, *U*, *V*, *Lx*, *Ly*, separate prin câte un spațiu;
- pe fiecare dintre următoarele *n* linii se află o pereche de numere X_i și Y_i, care reprezintă coordonatele unuia dintre pruncte ($U < X_i < U + Lx$, $V < Y_i < V + Ly$);

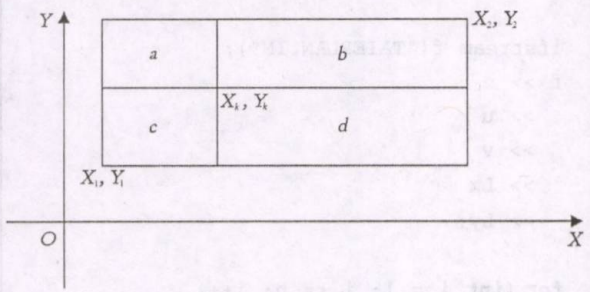
Date de ieșire

La ieșirea standard vor fi scrise coordonatele dreptunghiului găsit, precum și suprafața acestuia.

Indicație

Suprafața dreptunghiulară poate fi memorată în două moduri:

- coordonatele a două colțuri diagonal opuse; de exemplu: *Stânga-Jos* (x_1, y_1) și *Dreapta-Sus* (x_2, y_2);
- coordonatele (x_1, y_1) ale unui colț (de exemplu *Stânga-Jos*) și laturile dreptunghiului *Lx* și *Ly*.



Exemplu

TAIEPLAN.IN

```
3 4 6 26 34
6 10
15 25
25 35
```

Ieșire

```
x1f = 6
y1f = 4
x2f = 30
y2f = 25
Smax = 504
```

Soluție

În fiecare punct p_k se efectuează două tăieturi; pentru fiecare tăietură se obțin câte două noi dreptunghiuri pentru care se reia procedeul de tăiere după punctul p_{k+1}.

Pentru $k \le n$, dacă p_k este pe suprafața dreptunghiului curent (x_1, y_1, x_2, y_2), atunci se efectuează cele două tăieturi (se obțin patru noi dreptunghiuri).

Listing TAIEPLAN.CPP (I)

```
#include<fstream.h>
#include<iomanip.h>

typedef struct pct2
{
  int x, y;
}
Punct;

Punct p[100];
int n;
int x1, y1;
int x2, y2;
int x1f, y1f;
int x2f, y2f;
int Smax = 0;

void Citeste()
{
  int u, v;
  int Lx, Ly;

  ifstream f("TAIEPLAN.IN");
  f >> n
    >> u
    >> v
    >> Lx
    >> Ly;

  for (int i = 1; i <= n; i++)
  {
    f >> p[i].x >> p[i].y;
  }

  x1 = u;
  y1 = v;
  x2 = u + Lx;
  y2 = v + Ly;

  f.close();
}

int Inside(int k, int x1, int y1,
                              int x2, int y2)
{
  return ((x1 <= p[k].x) &&
          (p[k].x <= x2) &&
          (y1 <= p[k].y) &&
          (p[k].y <= y2))
}
```

Listing TAIEPLAN.CPP (II)

```
void RetineSolutia(int k, int x1, int y1,
                                  int x2, int y2)
{
  int s = (x2 - x1) * (y2 - y1);

  if (s > Smax)
  {
    Smax = s;
    x1f = x1;
    y1f = y1;
    x2f = x2;
    y2f = y2;
  }
}

void TaieDivEtImp(int k, int x1, int y1,
                                  int x2, int y2)
{
  if (k <= n)
  {
    if (Inside(k, x1, y1, x2, y2))
    {
      TaieDivEtImp(k + 1, x1, p[k].y, x2, y2);
      TaieDivEtImp(k + 1, x1, y1, x2, p[k].y);
      TaieDivEtImp(k + 1, x1, y1, p[k].x, y2);
      TaieDivEtImp(k + 1, p[k].x, y1, x2, y2);
    }
    else
    {
      TaieDivEtImp(k + 1, x1, y1, x2, y2);
    }
  }
  else
  {
    RetineSolutia(x1, y1, x2, y2);
  }
}

void main()
{
  Citeste();
  TaieDivEtImp(1, x1, y1, x2, y2);

  cout << "x1f = " << x1f << endl;
  cout << "y1f = " << y1f << endl;
  cout << "x2f = " << x2f << endl;
  cout << "y2f = " << y2f << endl;

  cout << "Smax = " << Smax << endl;
}
```

Pentru $k \le n$, dacă p_k este pe suprafața dreptunghiului curent (x_1, y_1, x_2, y_2), atunci se reapelează funcția cu același dreptunghi (x_1, y_1, x_2, y_2) pentru punctul p_{k+1}.

Pentru $k > n$ (s-au efectuat toate tăieturile) se reține soluția pentru dreptunghiul curent.

Problema tăieturilor în spațiu

Se dă un corp în formă de paralelipiped. Corpul conține în interiorul său n bule de aer punctiforme (cu dimensiuni neglijabile). Să se determine în interiorul acestuia un paralelipiped cu volumul maxim, având muchiile paralele cu muchiile paralelipipedului inițial.

Acest paralelipiped nu trebuie să conțină nicio bulă de aer în interior.

Pe muchii sau pe suprafețele laterale pot exista oricâte bule de aer.

Corpul este dat într-un sistem tridimensional de axe și are muchiile paralele cu axele de coordonate.

Date de intrare

Datele din fișierul de intrare **PARALEL.IN** sunt structurate astfel:

- pe prima linie se află șapte numere, separate prin spații:
  - numărul n al bulelor de aer;
  - coordonatele (U, V, W) ale celui mai apropiat vârf de origine;
  - cele trei dimensiuni (Lx, Ly, Lz) ale corpului paralelipipedic;
- pe fiecare dintre următoarele n linii se află un triplet de numere X_i și Y_i și Z_i, care reprezintă coordonatele unuia dintre puncte $(U < X_i < U + Lx, V < Y_i < V + Ly, W < Z_i < W + Lz)$;

Date de ieșire

La ieșirea standard vor fi scrise coordonatele paralelipipedului găsit, precum și suprafața acestuia.

Indicații

Suprafața dreptunghiulară poate fi memorată în două moduri:

- coordonatele a două colțuri diagonal opuse; de exemplu colțul cel mai apropiat de originea axelor de coordonate și colțul diagonal opus acestuia;
- coordonatele (x_1, y_1) ale unui colț (de exemplu colțul cel mai apropiat de originea axelor de coordonate) și laturile paralelipipedului Lx, Ly și Lz.

Soluție

Alegem prima variantă de memorare a paralelipipedului. Implementarea care folosește cea de-a doua variantă de memorare este propusă ca temă.

Folosim următoarele notații:

- X_1, Y_1, Z_1 reprezintă coordonatele celui mai apropiat vârf de origine;
- X_2, Y_2, Z_2 reprezintă coordonatele vârfului diagonal opus celui mai apropiat vârf de origine;

Listing PARALEL.CPP (I)

```
#include<fstream.h>
#include<iomanip.h>
typedef struct pct3
{
  int x, y, z;
} Punct;

Punct p[100];
int n;
int x1, y1, z1;
int x2,  y2, z2;
int x1f, y1f, z1f;
int Lxf, Lyf, Lzf;
int Vmax = 0;

void Citeste()
{
  int X, Y, Z;
  int Lx, Ly, Lz;
  ifstream f("PARALEL.IN");
  f >> n >> X >> Y >> Z >> Lx >> Ly >> Lz;
  for (int i = 1; i <= n; i++)
    f >> p[i].x >> p[i].y >> p[I].z;
  x1 = X;
  y1 = Y;
  z1 = Z;
  x2 = X + Lx;
  y2 = Y + Ly;
  z2 = Z + Lz;
  f.close();
}

int Inside(int k, int x1, int y1, int z1,
                  int x2, int y2, int z2)
{
  return ((x1 <= p[k].x) &&
          (p[k].x <= x2) &&
          (y1 <= p[k].y) &&
          (p[k].y <= y2) &&
          (z1 <= p[k].z) &&
          (p[k].z <= z2))
}

void RetineSolutia(int k, int x1, int y1,
           int z1, int x2, int y2, int z2)
{
  int vl = (x2 - x1) * (y2 - y1) * (z2 - z1);
  if (vl > Vmax)
  {
    Vmax = vl;
```

Listing PARALEL.CPP (II)

```
    x1f = x1;
    y1f = y1;
    z1f = z1;
    Lxf = x2 - x1;
    Lyf = y2 - y1;
    Lzf = z2 - z1;
  }
}

void TaieDivEtImp(int k, int x1, int y1,
            int z1, int x2, int y2, int z2)
{
  if (k <= n)
    if (Inside(k, x1, y1, z1, x2, y2, z2))
    {
      // plan z = p[k].z
      TaieDivEtImp(k + 1, x1, y1, p[k].z,
                                  x2, y2, z2);
      TaieDivEtImp(k + 1, x1, y1, z1,
                           x2, y2, p[k].z);

      // plan y = p[k].y
      TaieDivEtImp(k+1, x1, p[k].y, z1,
                                  x2, y2, z2);
      TaieDivEtImp(k+1, x1, y1, z1,
                           x2, p[k].y, z2);

      // plan x = p[k].x
      TaieDivEtImp(k+1, p[k].x, y1, z1,
                                  x2, y2, z2);
      TaieDivEtImp(k+1, x1, y1, z1,
                           p[k].x, y2, z2);
    }
    else
      TaieDivEtImp(k + 1, x1, y1, z1,
                                  x2, y2, z2);
  else
    RetineSolutia(x1, y1, z1, x2, y2, z2);
}

void main(){
  Citeste();
  TaieDivEtImp(1, x1, y1, z1, x2, y2, z2);
  cout << "x1f = " << x1f << endl;
  cout << "y1f = " << y1f << endl;
  cout << "z1f = " << z1f << endl;
  cout << "x2f = " << x2f << endl;
  cout << "y2f = " << y2f << endl;
  cout << "z2f = " << z2f << endl;
  cout << "Vmax = " << Vmax << endl;
}
```

- X_f, Y_f, Z_f reprezintă coordonatele celui mai apropiat vârf de origine al paralelipipedului determinat;
- Lx_f, Ly_f, Lz_f reprezintă dimensiunile paralelipipedului determinat.

Pentru fiecare punct p_k (care reprezintă o bulă de aer) se efectuează trei tăieturi:

- după un plan paralel cu XOY (coordonata z a planului este dată de coordonata z a punctului p_k);
- după un plan paralel cu XOZ (coordonata y a planului este dată de coordonata y a punctului p_k);
- după un plan paralel cu YOZ (coordonata x a planului este dată de coordonata x a punctului p_k).

La fiecare tăietură se obțin câte două noi paralelipipede pentru care se reia procedeul de tăiere după punctul p_{k+1}.

Rigleta cu pioni

Se consideră o rigletă care conține N căsuțe, numerotate da la 1 la N, de la dreapta spre stânga.

În orice moment, o căsuță poate conține cel mult un pion (valoarea 0 indică faptul că acea căsuță nu conține niciun pion, iar valoarea 1 indică faptul că acea căsuță conține exact un pion).

Depunerea/extragerea a N pioni în/din cele N căsuțe se realizează prin respectarea regulilor următoare:

- în orice moment se poate amplasa (dacă nu există) sau se poate ridica (dacă există) un pion în/din căsuța numerotată cu 1;
- în orice moment se poate amplasa (dacă nu există) sau ridica (dacă există) un pion în/din căsuța numerotată cu 2, dacă prima căsuță este ocupată ($R_1 = 1$);
- în orice moment se poate amplasa (dacă nu există) sau ridica (dacă există) un pion în/din căsuța numerotată cu k ($2 < k \leq N$), dacă există un pion în căsuța numerotată cu $k - 1$, și primele $k - 2$ căsuțe sunt libere;

Cerința A

Se consideră că rigleta are toate căsuțele libere; să se depună cei n pioni astfel:

- se definește și se afișează starea inițială a rigletei;
- se determină și se afișează stările intermediare ale rigletei;
- se determină și se afișează starea finală a rigletei.

Cerința B

Se consideră că rigleta are toate căsuțele ocupate, să se extragă cei n pioni astfel:

- se definește și se afișează starea inițială a rigletei;
- se determină și se afișează stările intermediare ale rigletei;
- se determină și se afișează starea finală a rigletei.

Indicații

Fie $R[1 .. n]$ un vector care conține numai valorile 0 și 1, definit astfel:

- $R_i = 1$ în cazul în care căsuța numerotată cu i conține un pion;

- $R_i = 0$ în cazul în care căsuța numerotată cu i nu conține niciun pion;

Operațiile descrise în enunț duc la generarea codului *Gray*:
- pentru cerința *A* se generează codul *Gray* în ordine crescătoare;
- pentru cerința *B* se generează codul *Gray* în ordine descrescătoare;

Soluție

Se construiesc funcțiile recursive:
- `DepunePrimele(k)` - amplasează pioni în fiecare dintre primele *k* căsuțe;
- `ExtragePrimele(k)` - ridică pionii din fiecare dintre primele *k* căsuțe;
- `DepunePozitia(k)` - amplasează un pion în căsuța numerotată cu *k* ($R_k = 1$);
- `ExtragePozitia(k)` - ridică pionul din căsuța numerotată cu *k* ($R_k = 0$).

Pentru $k \geq 1$ avem:

- `DepunePrimele(k) =`
  ```
  DepunePrimele(k - 1)
  ExtragePrimele(k - 2)
  DepunePozitia(k)
  DepunePrimele(k - 2)
  ```

- `ExtragePrimele(k) =`
  ```
  ExtragePrimele(k - 2)
  ExtragePozitia(k)
  DepunePrimele(k - 2)
  ExtragePrimele(k - 1)
  ```

Exemplu

Pentru $N = 4$ avem:

A. Depunere pioni

```
4 3 2 1
0  0  0  0
0  0  0  1   (1)
0  0  1  1   (2)
0  0  1  0   (1)
0  1  1  0   (3)
0  1  1  1   (1)      DepunePrimele(n - 1)
0  1  0  1   (2)
0  1  0  0   (1)      ExtragePrimele(n - 2)
1  1  0  0   (3)      DepunePozitia(n)
1  1  0  1   (1)
1  1  1  1   (2)      DepunePrimele(n - 2)
```

B. Extragere pioni

```
4 3 2 1
1  1  1  1
1  1  0  1   (2)
1  1  0  0   (1)      ExtragePrimele(n - 2)
0  1  0  0            ExtragePozitia(n)
0  1  0  1   (1)
0  1  1  1   (2)      DepunePrimele(n - 2)
0  1  1  0   (1)
0  0  1  0   (3)
0  0  1  1   (1)
0  0  0  1   (2)
0  0  0  0   (1)      ExtragePrimele(n - 1)
```

În casete este prezentată implementarea pentru cerința *A* (generarea codului *Gray* în ordine crescătore).

Listing RIGLETA.CPP (I)

```
#include<fstream.h>
#include<iomanip.h>

int n, R[100], ct=0;

void Scrie()
{
  cout << endl << setw(3) << ct << "=  ";
  for (int i = n; i >= 1; i--)
    cout << R[i];
  ct++;
}

void ExtragePrimele(int);

void DepunePrimele(int k)
{
  if (k >= 1)
  {
    DepunePrimele(k - 1);
    ExtragePrimele(k - 2);
    R[k] = 1;
```

Listing RIGLETA.CPP (II)

```
    Scrie();
    DepunePrimele(k - 2);
  }
}

void ExtragePrimele(int k)
{
  if (k >= 1)
  {
    ExtragePrimele(k - 2);
    R[k] = 0;
    DepunePrimele(k - 2);
    Scrie();
    ExtragePrimele(k - 1);
  }
}

void main()
{
  cout << endl << "n= ";
  cin >> n;
  memset(R, 0, sizeof(R));
  Scrie();
  DepunePrimele(n);
}
```

PARALELISM și CONCURENȚĂ în algoritmul BACKTRACKING RECURSIV

Tiberiu Socaciu

Variantele recursive ale majorității algoritmilor sunt modalități de exprimare simplă a clarității acțiunilor care urmează a fi efectuate, activități reduse cu un nivel de complexitate pentru anumite secvențe. Unul dintre cele mai clare exemple este cel al algoritmului backtracking, a cărui variantă recursivă este foarte ușor de ținut minte.

Subprogramul[1] recursiv pentru algoritmul *backtracking* este următorul[2]:

```
algoritm btr (level, np, p, x)
  dacă level = np + 1 atunci
    apelează prelucrează_soluție (np, x)
  altfel
    pentru i ← 1, p_level execută
      x_level ← i
      dacă fi (level, x) atunci
        apelează btr (level + 1, np, p, x)
      sfârșit dacă
  sfârșit pentru
  sfârșit dacă
sfârșit algoritm
```

Apelul subprogramului se realizează prin[3]:

```
apelează btr (1, np, p, x)
```

[1] *subprogramul este descris în pseudocod, parametrii se transmit prin valoare*

[2] *vezi [1], [2], [3], [10]*

[3] *pentru demonstrarea corectitudinii, vezi [1], [2], [3], [16]*

Paralelizarea algoritmului backtracking recursiv

Presupunem că avem la dispozitie un număr de *p* procesoare care ar putea prelua în paralel *task*-uri legate de algoritmul nostru.

Cea mai simplă metodă de a "sparge" domeniul dat în *p* subdomenii de dimensiuni asemănătoare constă în "spargerea" mulțimii {1, 2, ..., np_1} în *p* submulțimi disjuncte[4], care vor fi procesate în paralel[5] pentru înlocuirea apelului recursiv inițial:

```
pentru i ← 1, np_1 pas p execută
  în paralel
    secvențial
      x_1 ← i
      apelează btr (2, np, p, x)
    sfârșit secvențial
```

[4] *câte o submulțime pentru fiecare procesor*

[5] *prin* **în paralel** $i_1 | i_2 | \ldots | i_p$ **sfârșit în paralel** *am notat executarea instrucțiunilor $i_1, i_2, \ldots, i_p$ în paralel, pe cele p procesoare*

```
    secvențial
      x_1 ← i + 1
      apelează btr (2, np, p, x)
    sfârșit secvențial
  |
    ...
  |
    secvențial
      x_1 ← i + p - 1
      apelează btr (2, np, p, x)
    sfârșit secvențial
  sfârșit paralel
sfârșit pentru
```

Soluția anterioară va funcționa corect numai dacă valoarea np_1 este multiplu al valorii p. Evident, această condiție nu este întotdeauna îndeplinită; așadar, vom modifica soluția adăugând filtre pentru depășirea domeniului:

```
pentru i ← 1, np_1 pas p execută
  în paralel
    secvențial
      x_1 ← i
      dacă x_1 ≤ np_1 atunci
        apelează btr (2, np, p, x)
      sfârșit dacă
    sfârșit secvențial
  |
    secvențial
      x_1 ← i + 1
      dacă x_1 ≤ np_1 atunci
        apelează btr (2, np, p, x)
      sfârșit dacă
    sfârșit secvențial
  |
    ...
  |
    secvențial
      x_1 ← i + p - 1
      dacă x_1 ≤ np_1 atunci
        apelează btr (2, np, p, x)
      sfârșit dacă
    sfârșit secvențial
  sfârșit paralel
sfârșit pentru
```

Care sunt punctele slabe ale soluției de mai sus? Există situații[6] în care anumite procesoare vor fi "în șomaj", deoarece vor termina mai repede[7], în timp ce alte procesoare încă nu au terminat efectuarea operațiilor corespunzătoare. Datorită modului în care a fost implementat algoritmul, aceste procesoare nu vor primi noi *task*-uri[8].

[6] *altele decât procesarea celor np_1 % p elemente care rămân la final*

[7] *deoarece, din motive date de evaluarea funcției de continuare, se iau decizii de a nu continua avansarea*

[8] *se termină într-un număr finit de pași*

Se pot imagina scheme mai reușite de decompoziție și de parsare a domeniului[9], dar toate solutiile vor avea dezavantajul că există șanse ca unele procesoare să termine efectuarea operațiilor alocate mai repede și să nu primească noi sarcini.

Rezolvarea acestei probleme se poate realiza prin introducerea comunicării între procesoare[10] și o anumită rebalansare a activităților în cazul în care anumite procesoare devin disponibile[11].

Concurențialitatea

Concurențialitatea poate fi o soluție de implementare a algoritmului în momentul în care mașina pe care rulăm programul nostru nu este o mașină fizică, ci una virtuală (practic, un *cluster* de calculatoare[12]).

Mașina virtuală se ocupă de maparea proceselor nou create peste spațiul fizic de procese din *cluster*[13]. La prima vedere, soluția este clară: se genereaza np_i procese în paralel pentru fiecare valoare posibilă din x_i[14].

Descrierea în pseudocod a acestei variante este:

```
algoritm btr (level, np, p, x)
  dacă level = np + 1 atunci
    apelează prelucrează_soluție (np, x)
  altfel
    pentru i ← 1, p_level execută
      x_level ← i
      dacă fi (level, x) atunci
        dacă fork() = 0 atunci
          apelează btr (level + 1, np, p, x)
        sfârșit dacă
      sfârșit dacă
  sfârșit pentru
  sfârșit dacă
sfârșit algoritm
```

Cea mai importantă întrebare care se impune este: *ce se întâmplă cu numărul proceselor? Nu crește acest număr prea mult*[15]?

Limitarea numărului proceselor

Soluția ar consta în implementarea unui mecanism oricât de simplu de comunicare pentru a se lua decizia corectă cu privire la o potențială lansare în execuție a unui proces fiu concurent.

[9] *de exemplu, una dintre ele ar putea consta în împărțirea în p zone succesive de forma $[(k-1)\cdot a+1, (k-1)\cdot a+2, \ldots, k\cdot a]$*

[10] *mai exact, între procesele care rulează pe procesoare*

[11] *vezi [4] și [6] pentru algoritmul propus de autor în cazul unui cluster bazat pe mesaje*

[12] *de exemplu, vaiante de Mosix sau OpenMosix*

[13] *vom presupune că crearea unui nou proces se va reliza tot cu o primitivă de gen fork*

[14] *Am folosit fork (efectul este similar funcției omonime din C sub Unix)*

[15] *cu un fork-flood putem "ucide" orice sistem tip Unix în momentul în care spațiul de procese se umple și nu se pot aloca noi pid-uri proceselor care se solicită a fi create*

Să presupunem că avem o funcție `Decizie` care returnează o evaluare a oportunității de a lansa în execuție un nou fiu. Această funcție de evaluare ar putea număra procesele fiu deja în curs de rulare și, în cazul în care numărul proceselor nu este prea mare, ar putea permite crearea unui nou proces; în caz contrar totul se transformă într-un apel clasic recursiv.

```
algoritm btr (level, np, p, x)
  dacă level = np + 1 atunci
    apelează prelucrează_soluție (np, x)
  altfel
    pentru i ← 1, p_level execută
      x_level ← i
      dacă fi (level, x) atunci
        dacă Decizie() atunci
          dacă fork() = 0 atunci
            apelează btr (level + 1, np,
                                  p, x)
          sfârșit dacă
        altfel
          apelează btr (level + 1, np, p, x)
        sfârșit dacă
      sfârșit dacă
  sfârșit pentru
  sfârșit dacă
sfârșit algoritm
```

Este demn de remarcat faptul că funcția `Decizie` trebuie să fie foarte rapidă deoarece ea este apelată de foarte multe ori.

Exerciții și probleme

1. Modificați varianta recursiv-paralelă astfel încât să se lanseze de la început, în paralel, p procese care vor parsa în bucle pentru domeniile pentru x_1 de forma $\{(k-1)\cdot a+1, (k-1)\cdot a+2, \ldots, k\cdot a\}$.
2. Identificați limbaje care permit paralelizare cu instrumente echivalente structurii **în paralel**.
3. Folosiți ideile de decompoziție din [4] și [6] pentru a modifica varianta recursiv-paralelă astfel încât aceasta să ruleze pe un *cluster* bazat pe mesaje.
4. Care este diferența dintre *Mosix* și *OpenMosix*? Dar dintre *Linux* și *Mosix/OpenMosix*?
5. Orice program compilat sub *Linux* va rula fără probleme sub *Mosix*? De ce se preferă rularea sub *Mosix* a programelor "prolifice" în a crea procese fiu?
6. Unificați în varianta recursiv-concurentă funcțiile `Decizie` și `fi`, prin crearea unei funcții `fi` care returnează una dintre valorile -1, 0 sau 1. Modificați varianta recursiv-concurentă după unificarea funcțiilor.
7. Implementați o funcție `Decizie` care să permită cel mult zece procese fiu la un moment dat pentru implementarea recursiv-concurentă.

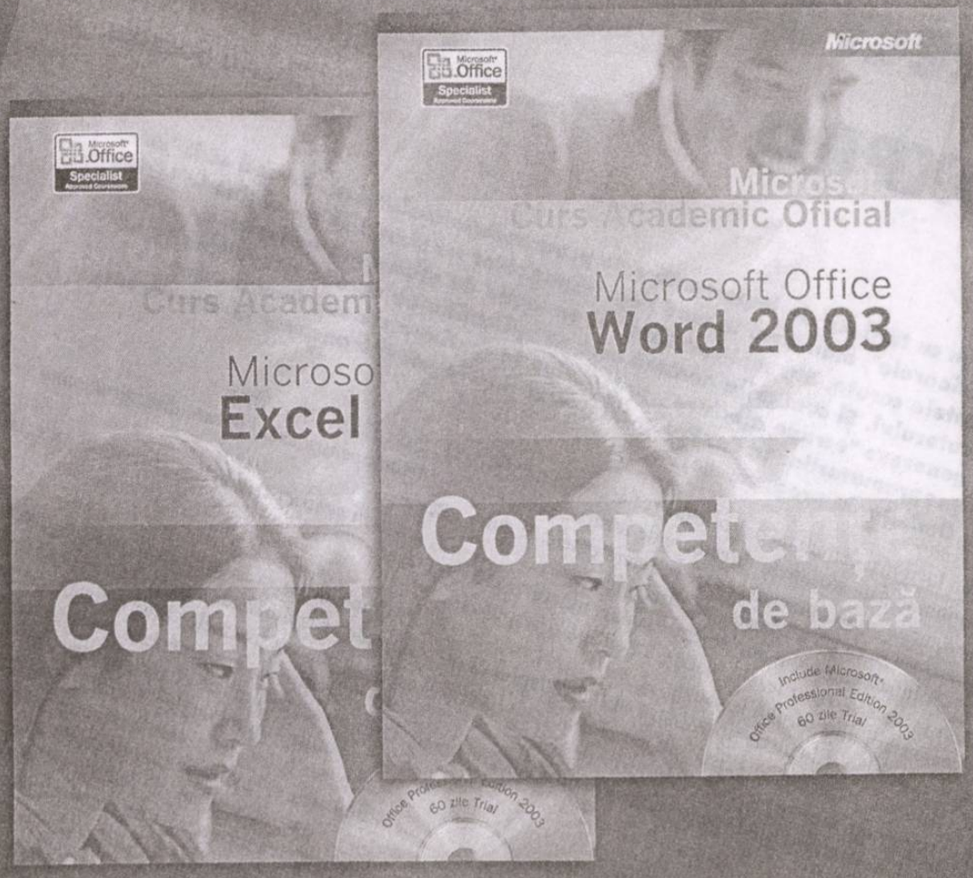
Bibliografie

1. Tiberiu Socaciu, *Învățați Backtracking. Exemple în Pascal și C*, Editura InfoData Cluj-Napoca, 2006, ISBN 973-87774-1-0, 160 de pagini;
2. Tiberiu Socaciu, *Metoda Backtracking cu exemple în Pascal*, Editura Edusoft Bacău, 2005, ISBN 973-87496-5-4, 100 de pagini;
3. Tiberiu Socaciu, *Metoda Backtracking cu exemple în C*, Editura Edusoft Bacău, 2005, ISBN 973-87496-6-2, 100 de pagini;
4. Bogdan Pătruț, Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, comunicare la Linux și mediile virtuale de instruire, Arad, 11-15 septembrie 2006, eveniment organizat de Universitatea de Vest "Vasile Goldiș" Arad;
5. Darius Bufnea, Tiberiu Socaciu, *De la backtracking secvențial la cel concurent și paralel*, comunicare susținută la Zilele Unirii, Focșani, 22-24 ianuarie 2005;
6. Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, GInfo nr. 16/8 (decembrie 2006), Editura Agora Media Târgu-Mureș;
7. Tiberiu Socaciu, *Cea mai bună funcție de continuare*, GInfo nr. 16/7 (noiembrie 2006), Editura Agora Media Târgu-Mureș;
8. Tiberiu Socaciu, *Serializarea ca metoda de determinare a funcției de continuare*, GInfo nr. 16/6 (octombrie 2006), Editura Agora Media Târgu-Mureș;
9. Tiberiu Socaciu, *O problemă de combinatorică*, GInfo nr. 11/4 (aprilie 2001), Editura Agora Media Târgu-Mureș;
10. Tiberiu Socaciu, *Metoda Backtracking*, if nr. 5/1992 (16), Editura Computer Press Agora Târgu-Mureș;
11. Bogdan Pătruț, Tiberiu Socaciu, *Algoritm Backtracking pentru un cluster bazat pe mesaje*, Studia Universitatis "Vasile Goldiș" Arad, seria Științe Exacte, nr. 16 (2006), în curs de apariție.
12. Tiberiu Socaciu, *Principiul contorului*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
13. Tiberiu Socaciu, *Rezolvarea unei probleme de backtracking*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
14. Tiberiu Socaciu, *Corectitudinea programelor*, teză de dizertație, Universitatea Babeș-Bolyai Cluj, Facultatea de Matematică și Informatică, 1996 (conducător științific: prof. univ. dr. Militon Frențiu);
15. Tiberiu Socaciu, *Demonstrarea corectitudinii algoritmilor care folosesc metoda backtracking*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
16. Tiberiu Socaciu, *Demonstrarea corectitudinii variantei recursive a algoritmilor care folosesc metoda backtracking*, GInfo nr. 17/5 (mai 2007), Editura Agora Media Târgu-Mureș (în curs de apariție).

NOU!

Microsoft®
Curs Academic Oficial

...acum, în limba română!



Manualele includ 2 CD-uri

Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Validitate...

Testarea DISCURILOR și DIRECTOARELOR în FOXPRO

Tiberiu Socaciu

Știm cu toții cât de importantă este programarea defensivă. Un cod scris "cu picioarele", manevrat în mod necorespunzător nu numai că nu obține la ieșire datele cerute, dar mai constituie și un motiv de etichetare generică a autorului. Și cum grija noastră este ca aplicațiile pe care le dezvoltăm să nu genereze "ecrane albastre", prezentăm în continuare două funcții utile programatorilor în FoxPro și nu numai (ideile pot fi transpuse în diverse alte limbaje de programare).

M-am lovit mai demult de o problemă reală: aveam nevoie de anumite date stocate în directoare de forma `D:\Date\aaaa`, unde `aaaa` este anul la care se referă datele. Și cum directorul în cauză nu exista, aplicația o lua razna...

Soluția a constat în implementarea unei functii `ValidDir()` care să returneze o valoare booleană corespunzătoare stării directorului în cauză.

Astfel, o valoare de tip `FALS` ne-ar indica faptul că directorul respectiv nu există sau nu este accesibil pentru scriere din motive legate de drepturile de acces.

O secvență posibilă pentru folosirea unei astfel de funcții ar putea fi:

```
If ValidDir ("D:\Date\2007") Then
   Acțiuni pentru cazul în care directorul există
Else
   Acțiuni pentru cazul în care directorul nu există
EndIf
```

O altă problemă de care m-am lovit a fost generarea unor date care să fie salvate pe dischetă.

Dar, oare, în momentul în care aș scrie datele pe dischetă, va fi aceasta introdusă în unitate? Va fi discheta citibilă?

Evident, o funcție `ValidDsk()` care să returneze o valoare booleană corespunzătoare stării unității *floppy* ar fi binevenită.

O secvență posibilă pentru folosirea unei astfel de funcții ar putea fi:

```
If ValidDsk("A") Then
   Acțiuni pentru cazul în care directorul există
Else
   Acțiuni pentru cazul în care directorul nu există
EndIf
```

Implementarea acestor funcții poate fi privită ca un exercițiu de programare de sistem (vezi [4]), ca un lucru util în implementarea sistemelor informatice (vezi [5], [6]) sau ca un exercițiu de implementare reală a legăturilor dintre *FoxPro* și alte limbaje de programare (vezi [1], [2], [3]).

Totuși, studiul implementărilor realizate în continuare ar putea fi util celor care întâmpină probleme similare în limbajele și mediile în care dezvoltă aplicații.

Toate secvențele de cod sunt autodocumentate, deci înțelegerea lor necesită un efort minim.

Funcția ValidDir

```
* Verifica daca exista un director
* Utilizare:
*    expl = ValidDir (expc)
*      ... = ... ValidDsk ("C:\cale\dir") ...
*         expl = rezultatul
*         expc = directorul care va fi testat
*
Function ValidDir
Parameter m.Director
Private m.Dummy
Private m.Fd
```

```
m.Dummy     = "InfoData.$$$"
m.Director  = AllTrim (m.Director)
m.Dummy     = m.Director + "\" + m.Dummy
m.Fd        = FCreate (m.Dummy)

If m.Fd >= 0
  = FClose (m.Fd)
EndIf

Delete File (m.Dummy)

Release m.Dummy

Return m.Fd > 0
```

Funcția ValidDsk

```
* verificare unitate floppy
*   expl = ValidDsk (expc [,expn])
*    ... = ... ValidDsk ("A:") ...
*        expl = rezultatul
*        expc = unitatea care va fi testata
*        expn = numarul de incercari
*                                  (implicit 3)
*
Function ValidDsk
Parameters m.DiskName, m.RetryNumber
Private DummyFile
Private m.TheDisk
Private BinProc
Private m.ForCounter

If Parameters () = 0
  Return .F.
EndIf

If Parameters () = 1
  RetryNumber = 3
EndIf

m.TheDisk = Upper (Left (m.DiskName, 1))

If Not (m.TheDisk $ "AB")
  Return .T.
EndIf

DummyFile = "C:\InfoData.$$$"

If File (DummyFile)
  Delete File &DummyFile
EndIf

BinProc = "DiskIO" + TheDisk
If .F.
  Do DiskIOA.Bin
  Do DiskIOB.Bin
EndIf

Load &BinProc

For ForCounter = 1 To RetryNumber
  Call &BinProc
  If File (DummyFile)
    Exit
  EndIf
EndFor

Release Module &BinProc

If Not File (DummyFile)
  Return .F.
EndIf

Delete File &DummyFile

Return .T.
```

Generarea bibliotecilor binare

```
rem generare biblioteci binare
rem
tasm diskioa
tlink diskioa
exe2bin diskioa.exe diskioa.bin
tasm diskiob
tlink diskiob
exe2bin diskiob.exe diskioa.bin
del *.map
del *.obj
del *.exe
```

Biblioteca DiskIOA

```
; DISKIOA.ASM
_text   segment byte public 'code'
        assume cs:_text,ds:_text
        org    0h
disk    equ    _A_

DiskIO  proc   far
        ;salvare registri
        push   ds
        push   es
        push   ax
        push   bx
        push   cx
        push   dx
        push   si
        push   di
        push   bp
        pushf
        ;ds:=cs
        push   cs      ;salvare pe stiva
        pop    ds      ;restaurare de pe stiva
        ;verifica floppy-ul
        mov    ah,0    ;functia de citire
```

```
        mov     dl,disk ;discul
        int     13h     ;apel
        ;daca cy = 1 atunci eroare
        jc      ko      ;eroare
        ;creaza fisierul
        ;c:\infodata.$$$
        mov     ah,3ch  ;functia creaza fisier
        mov     cx,0    ;atributul (normal)
        lea     dx,dummyc       ;ds:dx -> numele
                        ; fisierului (asciiz)
        int     21h     ;apel (ax = handle-ul)
        ;si inchide acest fisier
        mov     bx,ax   ;bx=handle
        mov     ah,3eh  ;functia inchide fisier
        int     21h     ;apel

ko:     ;restaurare registri
        popf
        pop     bp
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     es
        pop     ds
        ;revenire
        ret

_A_     equ     00h
_B_     equ     01h
dummyc  db      'c:\infodata.$$$',0

DiskIO  endp
_text   ends
end
```

Biblioteca DiskIOB

```
; DISKIOB.ASM
_text   segment byte public 'code'
        assume cs:_text,ds:_text
        org     0h
disk    equ     _B_

DiskIO  proc    far
        ;salvare registri
        push    ds
        push    es
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        push    bp
        pushf
        ;ds:=cs
        push    cs      ;salvare pe stiva
        pop     ds      ;restaurare de pe stiva
        ;verifica floppy-ul
        mov     ah,0    ;functia de citire
        mov     dl,disk ;discul
        int     13h     ;apel
        ;daca cy = 1 atunci eroare
        jc      ko      ;eroare
        ;creaza fisierul
        ;c:\infodata.$$$
        mov     ah,3ch  ;functia creaza fisier
        mov     cx,0    ;atributul (normal)
        lea     dx,dummyc       ;ds:dx -> numele
                        ; fisierului (asciiz)
        int     21h     ;apel (ax = handle-ul)
        ;si inchide acest fisier
        mov     bx,ax   ;bx=handle
        mov     ah,3eh  ;functia inchide fisier
        int     21h     ;apel

ko:     ;restaurare registri
        popf
        pop     bp
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     es
        pop     ds
        ;revenire
        ret

_A_     equ     00h
_B_     equ     01h
dummyc  db      'c:\infodata.$$$',0

DiskIO  endp
_text   ends
end
```

Bibliografie

1. Tiberiu Socaciu, *Demoni în FoxPro*, PC Report, Editura Computer Press Agora Târgu Mureș;
2. Tiberiu Socaciu, *Litere românești în FoxPro*, PC Report, Editura Computer Press Agora Târgu Mureș;
3. Tiberiu Socaciu, *Legarea modulelor C și Assembler cu FoxPro*, GInfo, Editura Agora Media Târgu Mureș;
4. Tiberiu Socaciu, *Programare de sistem*, manuscris
5. Tiberiu Socaciu, *O ... altfel de informatică*, Editura InfoData Cluj-Napoca;
6. Tiberiu Socaciu, *Informatica privită ... altfel*, Editura InfoData Cluj-Napoca.

Programare CONCURENTĂ și SEMAFOARE în FOXPRO

Tiberiu Socaciu

În cele ce urmează ne propunem să rezolvăm problema implementării în FoxPro a primitivelor pentru lucrul cu semafoare.

Semaforul este o variabilă întreagă de un tip special, care este folosită în controlul concurenței.

În principiu, zonele critice, cu acces limitat la numărul de procese, sunt implementate cu semafoare. Cel care a definit pentru prima dată noțiunea de semafor a fost *Dijkstra*[1].

Principalele operații care se execută asupra unui semafor sunt P și V, considerate atomice (adică odată ce operațiile au fost lansate, terminarea lor va avea loc numai în momentul în care acest lucru este permis, în tot acest timp procesul lansator "dormind", deci nefolosind resursele CPU[2]), respectiv *Wait* și *Signal*.

Cele două operații[3] incrementează, respectiv decrementează semaforul; trebuie remarcat faptul că operația de decrementare este blocată atâta timp cât prin decrementare valoarea semaforului nu mai este pozitivă.

Un caz particular de semafoare este cel al semafoarelor binare, care controlează eventualele accesuri exclusive la resurse (ex. imprimantă, discuri etc.)

Implementarea noastră se bazează pe o bază de date deschisă în mod partajat, care conține numele și valoarea fiecărui semafor.

Orice operație de modificare a valorii unui semafor (prin intermediul operațiilor P sau V) este realizată prin blocarea înregistrării corespunzătoare[4]: neblocarea acesteia ar putea avea efecte dezastruoase asupra integrității semafoarelor.

Operațiile implementate[5] în fișierul **SEMAFOR.PRG** sunt următoarele[6]:

- pornește modulul *Semafor*:
```
StartSemafor()
```
- oprește modulul *Semafor*:
```
StopBemafor()
```
- creează un nou semafor:
```
expl CreeazaSemafor(
        expc NumeVariabila,
        expn ValoareInitiala)
```
- citește valoarea unui semafor:
```
expn CitesteSemafor(
        expc NumeVariabila)
```
- operația P:
```
expl PSemafor(expc NumeVariabila)
```
- operația V:
```
expl VSemafor(expc NumeVariabila)
```
- operația *Wait*:
```
expl waitSemafor(expc NumeVariabila)
```
- operația *Signal*:
```
expl SignalSemafor(expc NumeVariabila)
```

Cea mai celebră problemă pe care se testează de obicei semafoarele este problema producătorului și a consumatorului.

Această problemă presupune că există un număr oarecare de procese *producător* (care generează elemente și le

[1] *vezi [DIJKSTRA]*
[2] *în mediile concurente pure*
[3] *P și V*
[4] *pentru a se asigura atomicitatea*
[5] *comentarii referitoare la metode de evitare și implementare a semafoarelor se pot găsi în [GEORGESCU]*
[6] *șablonul de apel este dat conform regulilor FoxPro*

amplasează pe o "bandă"), respectiv procese *consumator* (care preiau elementele de pe "bandă").

Cu ajutorul semafoarelor se poate controla foarte elegant impasul generat de subdepășire[7]. Pentru controlul depășirii benzii[8] va trebui folosit încă un semafor.

Exemplul constă în rularea ciclică a unuia sau mai multor procese producător[9] și a unuia sau mai mai multor procese consumator[10].

Rezultatele obținute prin interclasarea listelor furnizate de cele două procese[11] care rulează pe stații diferite sunt următoarele:

```
CONSUMATOR INAINTE 0
PRODUCATOR INAINTE 0
PRODUCATOR DUPA 1
CONSUMATOR DUPA 0
PRODUCATOR INAINTE 1
PRODUCATOR DUPA 1
PRODUCATOR INAINTE 1
PRODUCATOR DUPA 2
CONSUMATOR INAINTE 2
CONSUMATOR DUPA 3
PRODUCATOR INAINTE 3
PRODUCATOR DUPA 4
CONSUMATOR INAINTE 4
CONSUMATOR DUPA 3
CONSUMATOR INAINTE 3
CONSUMATOR DUPA 2
CONSUMATOR INAINTE 2
CONSUMATOR DUPA 1
PRODUCATOR INAINTE 1
PRODUCATOR DUPA 2
PRODUCATOR INAINTE 2
PRODUCATOR DUPA 3
etc.
```

Teme

1. Încercați să folosiți biblioteca pentru implementarea diverselor probleme clasice de concurență.
2. Folosiți biblioteca de semafoare pentru accesul exclusiv temporar la o bază de date cu anumite informații.

Bibliografie

[DIJKSTRA] E.W.Dijkstra, *Co-operating Sequential Processes.*

[GEORGESCU] Horea Georgescu, *Programare concurentă*, Editura Tehnică, 1996.

[SOCACIU] Tiberiu Socaciu, *Rafinament cu FoxPro*, Editura InfoData (în curs de apariție).

[7] încercarea de a prelua un element când "banda" nu conține niciun element (underflow)

[8] în cazul în care există o limitare superioară a capacității benzii (overflow).

[9] vezi PRODUCAT.PRG

[10] vezi CONSUMAT.PRG

[11] producător și consumator

Listing SEMAFOR.PRG (I)

```
* porneste modulul pentru semafoare
Procedure StartSemafor
Private BD
BD = Select(0)
Select 0
If Not File("Semafor.Dbf")
  Create Table "Semafor" (Variabila C(20),
                                  Valoare N(8))
  Index on Upper(AllTrim(Variabila))
                                  Tag Variabila
  Use
EndIf

Use "Semafor" Share
Set Order To Variabila
Select (BD)
Return

* opreste modulul pentru semafoare
Procedure StopSemator
Private BD
BD=Select(0)
Select "Semafor"
Use
Select(BD)
Return

* creeaza semafor
*   returneaza adevarat daca reuseste)
Function CreeazaSemafor
Parameter NumeVariabila, ValoareInitiala
Private BD
BD = Select(0)
Select "Semafor"
If Seek(Upper(AllTrim(NumeVariabila)))
  Select (BD)
  Return .F.
EndIf
Append Blank
=RLock()
Replace Variabila With NumeVariabila
Replace Valoare With ValoareInitiala
UnLock
Select (BD)
Return .T.
```

Listing SEMAFOR.PRG (II)

```
* returneaza valoarea semaforului
*    (-1 in caz de eroare)
Function CitesteSemafor
Parameter NumeVariabila
Private BD
BD=Select(0)
Select "Semafor"
Private Valoarea
If Seek(Upper(AllTrim(NumeVariabila)))
  Valoarea = Semafor.Valoare
Else
  Valoarea = -1
EndIf
Select (BD)
Return Valoarea

* operatia P
*    returneaza fals in caz de eroare
Function PSemafor
Parameter NumeVariabila
Private BD
BD = Select(0)
Select "Semafor"
If Not Seek(Upper(AllTrim(NumeVariabila)))
  Select (BD)
  Return .F.
EndIf
Private Indicator
Indicator=.T.
Do While Indicator
  =RLock()
  If Semafor.Valoare>0
    Replace Semafor.Valoare With
                          Semafor.Valoare - 1
    Indicator = .F.
  EndIf
  UnLock
EndDo
Select (BD)
Return .T.

* operatia V
*    returneaza fals in caz de eroare
Function VSemafor
Parameter NumeVariabila
Private ED
BD=Select(0)
Select "Semafor"
If Not Seek(Upper(AllTrim(NumeVariabila)))
  Select (BD)
  Return .F.
EndIf
=RLock()
Replace Semafor.Valoare With
                          Semafor.Valoare + 1
UnLock()
Select (BD)
Return .T.

* operatia Wait, identica cu operatia P
Function WaitSemafor
Parameter NumeVaxiabila
Return PSemafor(NumeVariabila)

* Operatia Signal, identica cu operatia V
Function SignalSemafor
Parameter NumaVariabila
Return VSemafor(NumeVariabila)
```

Listing PRODUCAT.PRG

```
Close All
Set Talk Off
Set Procedure To Semafor.Prg
Do StartSemafor
=CreeazaSamafor("Banda",0)
For I = 1 To 100
  Wait TimeOut Rand()
  ? "PRODUCATOR INAINTE: " +
       AllTrim(Str(CitesteSemafor("Banda"))))
  =VSemafor("Banda")
  ? "PRODUCATOR DUPA: " +
       AllTrim(Str(CitestaSemafor("Banda"))))
EndFor
```

Listing COSUMAT.PRG

```
Close All
Set Talk Off
Set Procedure To Semafor.Prg
Do StartSemafor
=CreeazaSamafor("Banda",0)
For I = 1 TO 100
  Wait TimeOut Rand()
  ? "CONSUMATOR INAINTE: " +
       AllTrim(Str(CitesteSemafor("Banda"))))
  =PSemafor("Banda")
  ? "CONSUMATOR DUPA: " +
       AllTrim(Str(CitesteSemafor("Banda"))))
EndFor
```

Biblioteca oamenilor de succes

Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro
BYBLOS

Model JS-PROTOTYPE pentru ALGORITMUL lui DIJKSTRA

Vlad Bazon

Paginile Web implică trei componente: conținut, aspect, funcționalitate. Se recomandă ca acestea să fie separate, una față de cealaltă. Elementele conținutului propriu-zis sunt marcate cu anumite etichete (tag-uri standard, sau identificatori introduși de autor); prin intermediul acestor etichete, elementele conținutului vor putea fi referite din modulele care gestionează atributele de aspect și din acelea care suportă interacțiunea cu utilizatorul sau extind posibilitățile browser-elor.

În caseta de pe pagina următoare sunt prezentate pe scurt principalele standarde *Web*.

Browser-ele moderne încorporează implementări (cu diverse diferențe) pentru *HTML*, *CSS*, *JavaScript* și *DOM*. Prntre altele, aceasta înseamnă că se poate folosi *JavaScript* pentru a accesa și a afișa structura *DOM* a unui document *Web*.

Există instrumente *DOM inspector* pentru diverse *browser*-e, care permit vizualizarea conținutul corespunzător unui document.

Un exemplu simplu cu Prototype

Prototype JavaScript Framework (http://www.prototypejs.org) oferă funcții pentru dezvoltarea de aplicații folosind *JavaScript*.

Vom realiza o mică aplicație: utilizatorul ar vrea să introducă numele și notele elevului și să obțină lista elevilor și a mediilor rezultate.

Începem cu o modelare sumară a părții *JavaScript*; după ce vom asigura funcționalitatea corectă a părții *JavaScript*, vom adăuga în pagină și celelalte elemente *HTML* necesare.

Presupunem că lucrăm în același director în care avem biblioteca `prototype.js`. Folosind un editor de text oarecare (poate fi și *Notepad*) creăm un fișier `test.html`, a cărui primă variantă este prezentată în casetă.

Listing test.html (versiunea #1)

```
<SCRIPT type = "text/javascript" src =
                                "prototype.js">
</SCRIPT>

<SCRIPT>
  var Elev = Class.create();
  Elev.prototype =
  {
    initialize: function(nume)
    {
      this.nume = nume;
    },
    media: function(note)
    {
      for (var sn = 0, i = 0, n = note.length;
                                  i < n; i++)
        sn += note[i];
      alert(this.nume + " " + sn / n);
    }
  }
  var unElev = new Elev(
                  "Nume-elev Prenume-elev");
  var note = new Array(6,8,7,9);
  unElev.media(note);
</SCRIPT>
```

Standarde Web

HTML
definește elementele standard de conținut ale unei pagini Web (de aici până aici este un paragraf; de aici până aici este un tabel, acestea sunt liniile sale, acestea sunt coloanele; de aici până aici este o listă care conține următoarele elemente; aici se va insera o imagine, având ca sursă fișierul specificat; de aici până aici este un script care trebuie executat într-un anumit moment etc.).

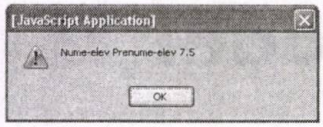
CSS
definește aspectul anumitor elemente HTML (toate paragrafele să arate astfel, iar un anumit paragraf trebuie afișat cu un anumit font și o anumită culoare; imaginile vor avea border:0 și vor fi poziționate la dreapta textului etc.).

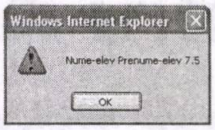
JavaScript
vizează comportamentul față de acțiunile utilizatorului asupra documentului (în momentul efectuării unui click pe un anumit buton, cere serverului să execute programul care generează o nouă imagine și inserează rezultatul drept nou conținut al elementului HTML identificat printr-un anumit ID; în momentul efectuării unui click pe un anumit buton inserează într-un anumit loc din document un anumit formular și setează cursorul într-un anumit câmp; în momentul efectuării unui click pe un anumit buton, execută o anumită funcție de verificare a valorilor introduse de utilizator în formular etc.).

DOM
este o interfață între HTML/CSS pe de o parte și JS pe de altă parte; DOM redefinește (față de HTML) structura logică a documentului, astfel încât să ușureze accesarea și modificarea dinamică a conținutului de către JS; DOM reprezintă conținutul unui document sub forma unui arbore în cadrul căruia fiecare nod reprezintă un tag HTML (sau, eventual, textul interior unui tag), iar legăturile între noduri sunt de tip child, parent sau sibling; fiecare nod reprezintă ca obiect un tag HTML - adică încorporează toate proprietățile și metodele specifice acelui obiect; elementele pot fi modificate dinamic folosind formule de acces de genul document. element.child.proprietate.

Putem deschide într-un *browser* fișierul **`test.html`** pentru a constata corectitudinea funcționării (dacă nu avem fișierul **`prototype.js`** atunci acesta trebuie descărcat și plasat în același director cu fișierul **`test.html`**).

Ar trebui să apară doar o casetă de dialog asemănătoare cu următoarele:





Prima imagine prezintă caseta de dialog generată de programul de navigare *Mozilla Firefox 2.0.0.3*, iar cea de-a doua o prezintă pe cea generată de *Windows Internet Explorer 7.0*.

Observație

Browser-ele moderne încorporează o implementare a limbajului *JavaScript*, dar nu neapărat aceeași; astfel "corectitudinea" unui script *JavaScript* depinde, eventual, și de *browser*-ul folosit și, de obicei, trebuie verificat pe mai multe *browser*-e (și eventual, ramificat corespunzător).

Explicații asupra scriptului

- primul element `<SCRIPT>` asigură includerea bibliotecii externe **`prototype.js`**;
- **`var`** asigură în *JavaScript* crearea unui obiect (variabilă);
- `Class` este un șablon de obiect ("prototip") introdus de biblioteca inclusă în fișierul **`prototype.js`**; în momentul instanțierii unui astfel de obiect (`var unElev = new Elev()`), funcția `create()` apelează automat funcția precizată prin `initialize`; ca urmare, obiectul particular `unElev` va încorpora (ca "date membru" cum am zice în *C++*) șirul `Nume-elev Prenume-elev`, accesibil din exterior prin `unElev.nume` și din interior prin `this.nume` (la fel ca în *C++*); `prototype` servește în *JavaScript* pentru extinderea unui obiect, permițând adăugarea dinamică a noi proprietăți sau metode.
- în *JavaScript* un obiect care reprezintă o colecție de obiecte (tabloul `note`) are o proprietate `length`, care reprezintă numărul actual de componente (aceasta este referită prin `Obiect.length`).

Observație

Am folosit

```
for (var sn = 0, i = 0, n = note.length;
                                    i < n; i++)
```

din același motiv pentru care în *C++* ar trebui să folosim:

```
for (i = 0, q = sqrt(n); i < q; i++)
                    // un singur apel sqrt(n)
```

în loc de:

```
for (i = 0; i < sqrt(n); i++)
                    // sqrt(n) este apelat
                    // la fiecare iterație!
```

Odată modelată funcționalitatea principală din partea *JavaScript*, trecem să adăugăm în **test.html** elementele necesare interacționării cu utilizatorul.

Desigur, va trebui să modificăm și partea *JavaScript*: deocamdată ne-am mulțumit cu `alert()` pentru a verifica funcționarea modulului *JavaScript* dar, de fapt, dorim să scriem rezultatele într-o listă creată în corpul documentului *HTML*.

Prevedem (minimal) următoarele elemente *HTML* pentru pagina pe care o vom crea:

- un formular cu două casete text (pentru nume și note) și cu un button care să reacționeze la evenimentul *click* (lansând funcția care preia datele, calculează media și să inserează rezultatul în listă);
- un element de tip listă, drept *container* pentru rezultatele furnizate pe parcursul interacțiunii cu utilizatorul.

Formularul și lista pot fi create folosindu-se următorul cod *HTML*:

```
<FORM action = "">
  Nume:
  <INPUT id = "nume"/>
  Note:
  <INPUT id = "note"/>
  <INPUT type = "button"
         value = "Adaugă în listă"
         onclick = "var elv = new
             Elev('nume'); elv.media('note')"
  />
</FORM>
<OL id = "lista">
</OL>
```

Observăm că în **onclick** nu mai transmitem direct numele și notele, ci transmitem *ID*-urile elementelor *HTML* în interiorul cărora utilizatorul va înscrie valorile respective (**ID** identifică unic în *DOM*, nodul respectiv - analog pointerilor din *C*).

DOM pune la dispoziție funcția `getElementById(ID_element)` cu ajutorul căreia se poate determina conținutul și proprietățile obiectului (*nodului*) corespunzător *ID*-ului primit ca parametru.

Prototype.js oferă un *shortcut* pentru `document.getElementById()` sub forma `$('id-element')`.

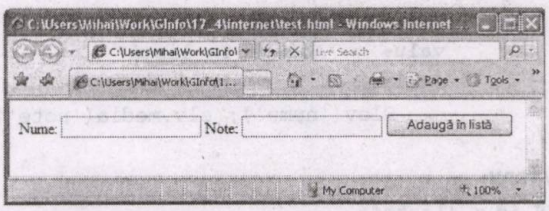
Nu toate browser-ele implementează `document.getElementById()`, dar `$()` este astfel concepută încât ramifică acțiunea corectă și pentru aceste *browser*-e.

Rămâne să modificăm corespunzător partea de *JavaScript* cu care am început. O singură problemă ar mai fi de remarcat: notele pe care le va introduce utilizatorul ca valoare a elementului identificat prin `note`, vor fi transferate către *JavaScript* ca text (șir de caractere, inclusiv spațiile separatoare; în general, prin *Internet*, se transferă șiruri de caractere, nu numere!); *JavaScript* oferă funcția `split()` pentru despărțire la spațiu (sau, la alt caracter precizat) a unui șir de caractere (rezultând un `Array` de șiruri de caractere) și, de asemenea, funcția de conversie necesară: `parseInt(string)`.

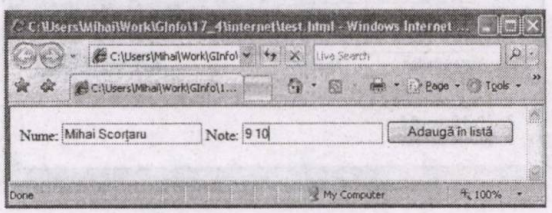
De exemplu `"7 10 8 9".split(" ");` va returna tabloul de șiruri de caractere `["7","10","8","9"];` însă, pentru a preveni situația în care utilizatorul separă notele prin mai multe spații (iar spațiul poate fi de multe feluri, inclusiv în informatică!) vom folosi `$('note').split(/\s+/);` unde expresia `/\s+/` desemnează în *JavaScript* (și în alte limbaje) orice succesiune formată din unul sau mai multe spații.

În final, întregul fișier **test.html** (dar încă nestilat prin *CSS*) este cel din caseta de pe pagina următoare.

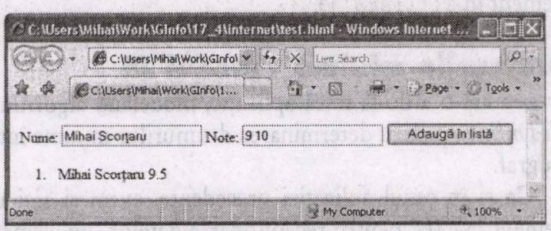
Dacă încărcăm pagina creată în programul de navigare *Windows Internet Explorer 7.0*, obținem următoarea pagină *Web*:



Putem începe să introducem date, așa cum se arată în continuare:



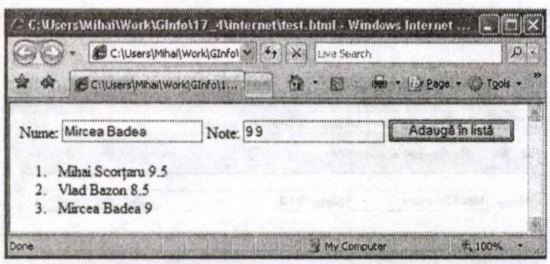
Dacă apăsăm butonul ***Adaugă în listă***, numele și media notelor introduse vor apărea în pagină:



Adaugând noi nume și note și apăsând butonul ***Adaugă în listă***, vom obține o listă formată din mai multe nume.

Listing test.html (versiunea #2)

```
<SCRIPT type = "text/javascript" src =
                                "prototype.js">
</SCRIPT>
<SCRIPT>
  var Elev = Class.create();
  Elev.prototype =
  {
    initialize: function(nume)
    {
      this.nume = $(nume).value;
    },
    media: function(note)
    {
      var note = $(note).value.split(/\s+/);
      for (var sn = 0, i = 0,
               n = note.length; i < n; i++)
        sn += parseInt(note[i]);
      $('lista').innerHTML += "<LI>" +
        this.nume + " " + sn / n + "</LI>";
    }
  }
</SCRIPT>
<FORM action = "">
  Nume:
  <INPUT id = "nume"/>
  Note:
  <INPUT id = "note"/>
  <INPUT type = "button"
         value = "Adaugă în listă"
         onclick = "var elv = new
             Elev('nume'); elv.media('note')"
  />
</FORM>
<OL id = "lista">
</OL>
```



Funcția `innerHTML()` este recunoscută de *browser*-ele moderne și permite modificarea conținutului unui nod *DOM* precizat (în cazul nostru - prin adăugarea unui nou element în `$('lista')`).

Drumurile minime - model JS

Prezentăm o a doua mică aplicație, modelând algoritmul lui *Dijkstra* pentru determinarea drumurilor minime într-un graf.

Ca și în cazul aplicației precedente, avem și aici un element *HTML* pentru recepționarea datelor (un element **<TEXTAREA>** în care utilizatorul va putea introduce matricea costurilor) și un element **<DIV>** (cu **id = 'result'**) în care *JavaScript* va insera rezultatele (drumurile minime). În plus, de data aceasta, am separat componentele nu numai în privința elaborării aplicației, ci chiar în fișiere distincte (**drumuri.html**, **dijkstra.js** și **layout.css**).

Listing drumuri.html

```
<HEAD>
  <META http-equiv = "Content-Type"
        content = "text/html;charset=utf-8">
  <META http-equiv = "Content-Script-Type"
        content = "text/javascript">
  <SCRIPT type = "text/javascript"
          src = "prototype.js">
  </SCRIPT>
  <SCRIPT type = "text/javascript"
          src = "dijkstra.js">
  </SCRIPT>
  <LINK rel = "stylesheet"
        href = "layout.css"
        type="text/css">
</HEAD>

<DIV id = "pagina">
  <FORM action = "">
    <FIELDSET>
      <LEGEND>
        Matricea costurilor
      </LEGEND>
      <P>Costurile sunt numere naturale;
        folosiți caracterul <B>x</B> pentru
        cazul în care <B>nu</B> există
        arc.</P>
      <P>Numerotați vârfurile astfel încât
        <B>1</B> să fie vârful sursă
        dorit.</P>
      <TEXTAREA id = "igraf"
                rows = "10"
                cols = "30">
      </TEXTAREA>
      <BR>
      <INPUT type = "button"
             class ="inpu"
             value = "Send"
             onclick = "$('result').
                              innerHTML = '';
                        var gr = new
                              wgraf('igraf');
                        gr.dijks(1);
                        for (var i = 2,
                             n = gr.nvf;
                             i <= n;
                             i++)
                          gr.wrpath(i)"
      />
      <P>Afișează distanțele și drumurile
        minime de la vârful <B>1</B>.</P>
    </FIELDSET>
  </FORM>

  <DIV id = "result">
  </DIV>
</DIV>
```

Observați cum reacționează la **onclick** butonul numit *Send*: șterge conținutul existent la `result`; apoi, construiește un obiect `gr` instanță a clasei `wgraf` (acesta corespunde grafului a cărui matrice a costurilor este trimisă părții *JavaSscript*) și apelează metoda `dijks()` pentru acest obiect; apoi, folosește metoda `wrpath()` a obiectului `gr`, în mod repetat, pentru toate vârfurile, ceea ce are ca rezultat scrierea în `result` a drumurilor găsite.

Listing dijkstra.js (I)

```
var infi = 100000;
var wgraf = Class.create();

wgraf.prototype =
{
  initialize: function(element)
  {
    var str = $(element).value
                .replace(/^\s+/mg, "")
                .replace(/\s+$/mg, "");
    var w = str.split(/[\n\r]+/g);
    this.nvf = w.length;
    this.cost = new Array();
    for(var i = 0, n = this.nvf; i < n; i++)
    {
      this.cost[i] = w[i].split(/\s+/);
      for (var j = 0; j < n; j++)
      {
        if (this.cost[i][j] == 'x')
          this.cost[i][j] = infi;
        this.cost[i][j] =
                    parseInt(this.cost[i][j]);
      }
    }
    this.dist = new Array(this.nvf);
    this.prev = new Array();
    this.done = new Array();
    for (var i = 0, n = this.nvf; i<n; i++)
    {
      this.dist[i] = parseInt(infi);
      this.done[i] = false;
      this.prev[i] = -1;
    }
  },

  dijks: function(from)
  {
    this.dist[from-1] = 0;
    var v, i, n = this.nvf;
    for (v = 0; v < n; v++)
    {
      var minw = infi, closest;
      for (i = 0; i < n; i++)
      {
        if (!this.done[i])
        {
          if (this.dist[i] < minw)
          {
            minw = this.dist[i];
            closest = i;
          }
        }
      }
      this.done[closest] = true;
      for (i = 0; i < n; i++)
      {
        if(!this.done[i])
        {
          var w = this.cost[closest][i];
          if (this.dist[closest]+w <
                                this.dist[i])
          {
            this.dist[i] =
                      this.dist[closest] + w;
            this.prev[i] = closest;
          }
        }
      }
    }
  },
```

Listing dijkstra.js (II)

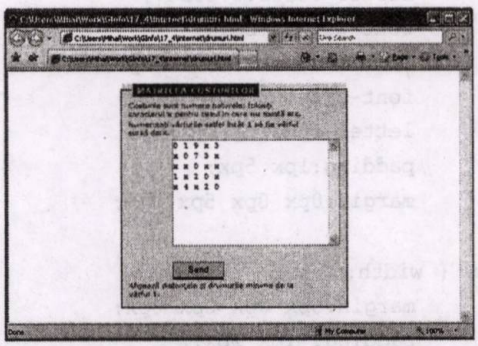
```
  wrpath: function(to)
  {
    var v = to - 1;
    var res = $('result');
    if (this.dist[v]<infi)
    {
      var path = "" + to;
      while (v > 0)
      {
        v = this.prev[v];
        if (v >= 0)
        {
          path = ""+(v+1)+" -> "+path;
        }
      }
      res.innerHTML +=
        "<P>distanța minimă = " +
        this.dist[to - 1] +
        " (" + path + ")</p>";
    }
    else
    {
      res.innerHTML =
        "<p>Nu există drum la vârful " +
                                  to+".</p>";
    }
  }
}
```

Note explicative

Să presupunem că utilizatorul a încărcat în *browser* fișierul **drumuri.html** și a obținut pagina următoare:

În continuare, a scris (de exemplu, folosind *Copy & Paste*) în elementul **<textarea>** al formularului matricea costurilor unui graf, de exemplu:

```
0 1 9 x 3
x 0 7 3 x
x x 0 x x
1 x 2 0 x
x 4 x 2 0
```



Apoi, utilizatorul a efectuat un click pe butonul *Send*. Matricea de mai sus arată "onorabil", dar uneori mai scapă niște spații inițiale, finale, sau între valori.

Evenimentul **onclick** asociat butonului *Send* determină crearea obiectului `gr` (`var gr = new wgraf(`

Listing layout.css

```
@charset "utf-8";
* { margin:0;padding:0; }
            /* afectează toate elementele */
body { background-color:#fff;color:#000;
       font-size:80%;
       font-family:Verdana,Arial,sans-serif;
}
#pagina { margin-left:25%; }
         /* pentru elementul cu ID=pagina */
p { font-size:1em;line-height:1.2em;
    color:#444;
} /* pentru toate paragrafele (inițial) */
fieldset { background-color:#fff9e7;
           font-size:1.2em;
           border-width:2px;
           border-style:solid;
           border-color:#7c5b47;
           margin:20px 0px 20px 0px;
           padding: 0px 10px 10px 10px;
           width:200px;
           position:relative;
           display:block;
}
fieldset p { font-size:0.7em;
             color:#448;
             text-align:left;
             margin-bottom:4px;
             margin-top:2px;
}
legend { position:relative;
         background-color:#7c5b47;
         color:#ffcc99;
         border-width:1px;
         border-style:solid;
         border-color:#7c5b47;
         font-weight:bold;
         font-variant:small-caps;
         font-size:0.9em;
         letter-spacing:1px;
         padding:1px 5px;
         margin:0px 0px 5px 0px;
}
.inpu { width: 80px;
         margin:5px 0px 0px 0px;
         padding: 1px 2px;
         border-width: 1px 1px 4px 1px;
         border-color:#657887;
         border-style: solid solid ridge solid;
         cursor:pointer;
}
```

'igraf');); așadar, se execută funcția precizată la `initialize` (cu parametrul `element = 'igraf'`); variabila temporară (locală) `str` preia valoarea conținută de nodul indicat prin `igraf` și elimină (aplicând `replace()`) eventualele spații inițiale sau finale din acest șir de caractere.

În continuare, `str` este despărțit (folosind `split()`) la caracterele de sfârșit de linie, rezultând tabloul temporar `w` (pentru matricea de mai sus, `w = [["0", "1", "9", "x", "3"], ["x", "0", "7", "3", "x"], ...]`).

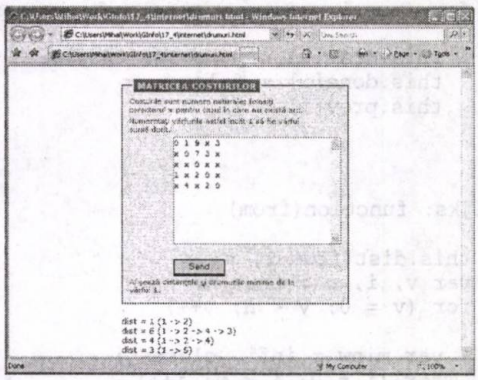
Pe baza matricei `w` se creează apoi variabilele membre `this.nvf` și `this.cost`; la crearea matricei cost a trebuit să înlocuim `"x"` cu infinit și să folosim `parseInt()` pentru a converti șirurile din `w` la valori numerice.

În continuare se creează și tablourile (date membre ale obiectului `gr`) `dist[]`, `prev[]`, `done[]` care sunt necesare metodelor `dijks()` și `wrpath()` ale obiectului `gr`.

Metoda `gr.dijks()` imită vizibil versiunea obișnuită în C++ a algoritmului lui *Dijkstra*.

Metoda `gr.wrpath()` folosește `gr.prev[]` pentru a adăuga în containerul `$('result')` drumul minim până la vârful primit ca parametru (având ca sursă vârful 1).

Rezultatul final este prezentat în următoarea imagine:



Fișierul **`layout.css`** încheie lucrul, propunând o variantă de stilare a elementelor *HTML* (cu împrumuturi, de exemplu pentru **<FORM>**, din diverse resurse publice)

Concluzie

Am folosit aici doar câteva elemente de bază puse la dispoziție de **`prototype.js`** (totuși suficiente pentru a putea aborda în maniera prezentată diverse alte aplicații, de exemplu dintre cele făcute inițial în C++).

Fișierul **`prototype.js`** are o dimensiune de aproximativ 70 KiB și va împlini în luna mai abia doi ani de la lansare.

S-a consacrat definitiv prin faptul că oferă platformă *cross-browser* de dezvoltare a aplicațiilor (extinzând *JavaScript* și în sensul rezolvării unor incompatibilități dintre *browser*-e), este comod de folosit (nu numai datorită "scurtăturilor" introduse pentru accesarea *DOM*-ului, dar și diverselor facilități adăugate la nivelul limbajului) dar mai ales datorită suportului excelent pe care îl oferă pentru tehnologia *AJAX*.

Construirea CUNOAȘTERII (II)

Trecerea de la reproducerea cunoașterii la construirea cunoașterii, utilizând soluții de eLearning

Profesorii, la fel ca și elevii, trebuie să se adapteze noilor tendințe. Cadrelor didactice li se cere să-și formeze noi competențe și abilități, iar o parte din acestea corespund unor noi profiluri de formare și unei profesii didactice care capătă noi valențe. Noțiunea de competențe distribuite și colective va dobândi o importanță sporită în contextul în schimbare al sistemelor de educație și formare. Programul SEI -Sistem Educațional Informatizat, încurajează profesorii, managerii școlari și personalul tehnic să se concentreze asupra fundamentelor noilor metode și tehnici de învățare: filosofii și abordări noi (constructivism, cognitivism, stiluri de învățare, inteligențe multiple), designul instruirii, proiectarea materialelor suport, strategii mai eficiente pentru atingerea obiectivelor învățării etc.

Societatea cunoașterii a impus utilizarea TIC ca punct de referință al schimbărilor de fond în sistemele de învățământ, iar provocările s-au conturat, la început, pe trei direcții majore, având ca punct de plecare summit-urile de la Lisabona și Feira: dotarea școlilor, pregătirea cadrelor didactice și asigurarea resurselor necesare. Ulterior, conjugându-se cu alte direcții prioritare, cum ar fi asigurarea calității în educație, învățarea pe parcursul întregii vieți, profesionalizarea meseriei didactice, aceste trei direcții au fost detaliate și completate. Acum se vorbește tot mai mult de o accelerare a procesului, însă premisele se modifică sensibil, prin prisma necesității de fundamentare teoretică psihologică și pedagogică, spre formarea de competențe transferabile și construirea de structuri și operații mentale, spre o regândire a strategiilor de predare-învățare și a sistemului de educație în sine.

O nevoie cheie, identificată de cele mai multe dintre studiile românești din domeniul educației, este lipsa de programe de instruire continuă adecvate, care să le permită profesorilor să-și formeze competențe cheie pentru activitățile didactice de tip nou. Aceste competențe includ:

(1) cunoștințe și abilități generale de utilizare a calculatorului, procesare de text, cunoștințe generale de comunicare, inclusiv e-mail și conectarea la Internet, proceduri generale de căutare pe web, instrumente de lucru online, inclusiv FTP, căutări în arhive online, proceduri de download, precum și utilizarea software-ului potrivit pentru toate aceste funcții.

(2) cunoștințe și competențe specifice, de utilizare a computerului pentru predare-învățare-evaluare.

În vederea respectării acestor noi cerințe, programul SEI oferă în primul rând o formare de tip general, prin cursuri în școală, dublată de un tip de pregătire hands-on a profesorilor, prin care sunt vizate:

- Cunoștințe teoretice și abilitățile necesare pentru utilizarea computerului, din perspectiva modificărilor tehnologice cu impact în desfășurarea activităților curente;
- Competențe necesare pentru integrarea noilor tehnologii ale informației și comunicării în activități didactice.

În al doilea rând, dincolo de utilizarea noilor tehnologii pentru predare-învățare-evaluare, este vizată familiarizarea cadrelor didactice cu computerul și Internetul ca instrumente necesare pentru autoformare și informare, complementare programelor de pregătire continuă. Se urmărește astfel, implicit, dezvoltarea unei culturi informatice înțeleasă nu numai ca o cunoaștere și competență de specialitate, ci și ca o nouă orientare și raportare la realitate. Astăzi, mai mult ca oricând, desfășurarea activității didactice ține de însușirea regulilor unui social în continuă dezvoltare și inovare.

Introducerea TIC în școli va contribui totodată la creșterea atractivității procesului de învățare, diversificarea metodelor și practicilor didactice și la deplasarea accentului de la caracterul informativ la cel formativ. Prin construirea de competențe specifice, elevii și profesorii vor avea posibilitatea să contribuie în mod activ la valoarea adăugată a procesului educațional și să își dezvolte criterii personale de selectare și utilizare a informației.

Olimpius Istrate,
Radu Jugureanu, Florin Ilia

DIRERASE sau cum să ștergem REPEDE

Tiberiu Socaciu

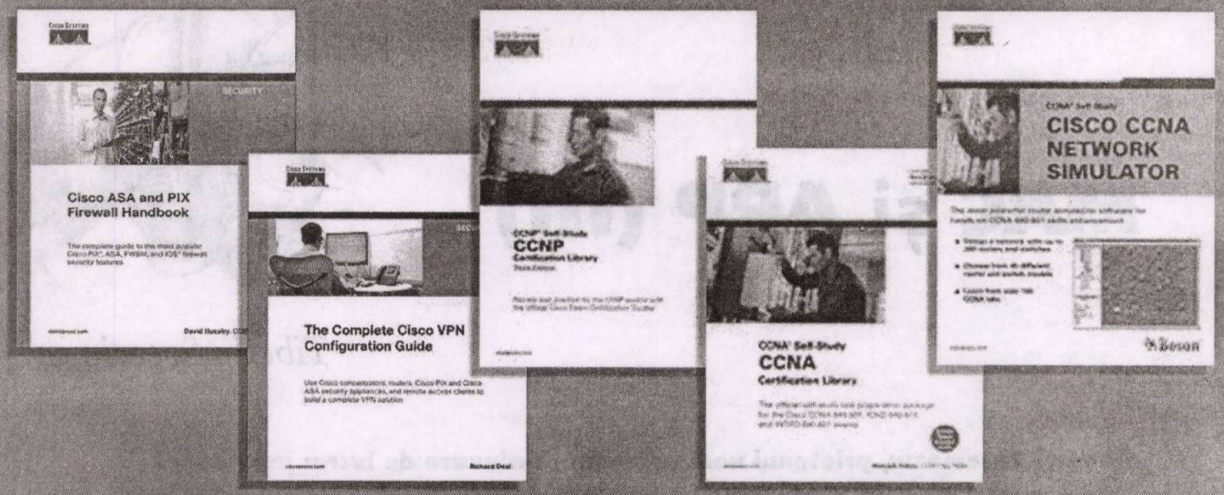
Ca o alternativă la comanda pentru ștergere în adâncime (XDEL, DELTREE etc.) prezentăm în continuare un program de ștergere a unei întregi ramuri din structura de directoare dintr-un sistem de fișiere organizat ierarhic. Programul se numește DIRERASE și la apel trebuie respectată sintaxa: DIRERASE <cale>.

```
Program DirErase;
uses dos;
const noerror = 0;
var f: File;
var path: String;
var l: Byte absolute path;
var i: Byte;

procedure del(path: string);
  var sr: Searchrec;
begin
  findfirst (path + '\*.*', anyfile, sr);
  while doserror = noerror do
    with sr do
    begin
      if (attr and directory = directory)
        and (name <> '.') and (name <> '..')
                                    then
    begin
      del (path + '\' + name)
      {$i-}
      Rmdir (path + '\' + name)
      {$i+}
      if ioresult = noerror then
        Write('Deleting directory ')
      else
        Write('Cannot delete directory ');
      Writeln (path + '\' + name)
    end
    else
  if (attr and directory <>
        directory) and (attr and volumeid
                  <> volumeid) then
    begin
      {$i-}
      Assign(f, path + '\' + name);
      Erase(f);
      {$i+}
      if ioresult = noerror then
        Write('Deleting file ')
      else
        Write('Can not delete file ');
      Writeln (path + '\' + name)
    end;
    Findnext(sr)
  end
end;

Begin
  if paramcount = 1 then
  begin
    Writeln ('Running DIRERASE...');
    path := Paramstr(1);
    if path [1] = '\' then
      Dec(l);
    for i := 1 to l do
    path[i] := Upcase(path[i]);
    {$i-}
    Rmdir(path);
    {$i+}
    if ioresult = noerror then
    Writeln('Deleting directory ', path)
  end
  else
    Writeln ('Usage: DIRERASE <path>')
End.
```

ISBN: 1587052156
Cisco ASA and PIX Firewall Handbook
Published: Dec 12, 2005

ISBN: 1587201046
CCNP Certification Library
Published: Dec 12, 2003

ISBN: 1587201313
Cisco CCNA Network Simulator
Published: Jul 7, 2004



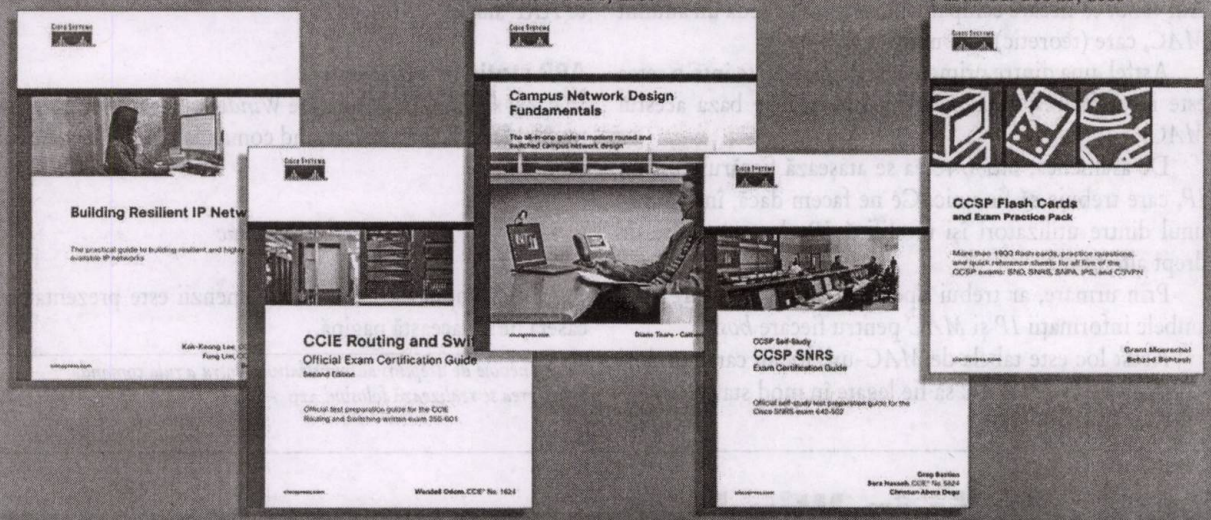
ISBN: 1587052040
The Complete Cisco VPN Configuration Guide
Published: Dec 15, 2005

ISBN: 1587200953
CCNA Certification Library
(CCNA Self-Study, exam #640-801)
Published: Aug 12, 2003

ISBN: 1587052156
Building Resilient IP Networks
Published: Dec 12, 2005

ISBN: 1587052229
Campus Network Design Fundamentals
Published: Dec 8, 2005

ISBN: 158720133
CCSP Flash Cards and Exam Practice Pack
Published: Dec 20, 2005



ISBN: 1587201410
CCIE Routing and Switching Official Exam Certification Guide
Published: Feb 17, 2006

ISBN: 1587201534
CCSP SNRS Exam Certification Guide (Exam Certification Guide)
Published: Dec 8, 2005

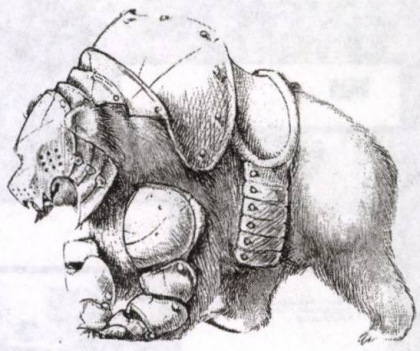
Învățătura e o treabă serioasă...

Investește deștept: CISCO PRESS

Cisco Systems
Cisco Press

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499



MAC și ARP (III)

Tiberiu Socaciu

Domnul Sistemescu, prietenul nostru, are în continuare de lucru: învață mai departe. Are aceleași probleme în rețea și știe că, dacă citește despre MAC-uri și ARP-uri, le poate rezolva singur, chiar dacă pe calculatoarele din rețea sunt, în continuare, activate filtre de tip firewall. De data aceasta va studia protecția la nivel de MAC...

ARP static ca metodă de securitate

Am văzut că fiecare echipament de rețea poseda un anumit *MAC*, care (teoretic) este unic.

Astfel, una dintre primele idei de securitate într-o rețea este identificarea unui membru în rețea pe baza acestui *MAC*.

De asemenea, într-o rețea se atașează fiecărui *host* un *IP*, care trebuie să fie unic. Ce ne facem dacă, în schimb, unul dintre utilizatori își modifică *IP*-ul pentru a se da drept altcineva?

Prin urmare, ar trebui operat în acel loc în care apar ambele informații *IP* și *MAC* pentru fiecare *host*.

Acest loc este tabela de *MAC*-uri, loc în care am dori ca informațiile *IP-MAC* să fie legate în mod static.

Metoda utilizată pentru a realiza acest lucru se numește *ARP static*.

ARP static în Windows

În cazul sistemului de operare *Windows*, legarea unui *IP* de un *MAC* se realizează folosind comanda `arp`[1] în următorul mod[2]:

```
arp -s ip mac
```

Un exemplu de utilizare a comenzii este prezentat în caseta de pe această pagină.

[1] *aveți nevoie de drepturi administrative pentru a rula comanda*
[2] *ștergerea se realizează folosind* `arp -d ip`

ARP static în Windows

```
C:\Documents and Settings\root>arp -a

Interface: 86.104.31.77 --- 0x2
  Internet Address      Physical Address      Type
  86.104.31.78          00-90-27-9f-ca-ea     dynamic

C:\Documents and Settings\root>arp -s 86.104.31.70 00-11-22-33-44-55

C:\Documents and Settings\root>arp -a

Interface: 86.104.31.77 --- 0x2
  Internet Address      Physical Address      Type
  86.104.31.70          00-11-22-33-44-55     static
  86.104.31.78          00-90-27-9f-ca-ea     dynamic
```

ARP static în Windows

```
[root@server t]# ifconfig eth1:100 86.104.30.254 netmask 255.255.255.0
[root@server t]# cat ethers
86.104.30.2      00:04:61:56:2A:9A
86.104.30.3      00:50:04:31:AA:FD
86.104.30.4      00:03:0D:0D:FC:34
86.104.30.5      00:02:44:6D:89:54
86.104.30.6      08:00:46:E8:E8:B7
86.104.30.7      00:06:1B:DF:71:0B
86.104.30.8      00:0A:E6:B7:CD:C9
86.104.30.9      00:0B:CD:AB:88:B8
86.104.30.10     00:11:2F:40:06:5A
[root@server t]# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
86.104.31.74             ether   00:40:F4:B3:8C:96   C                     eth1
86:104.31.77             ether   00:30:05:29:60:9A   C                     eth1
[root@server t]# arp -f ethers
[root@server t]# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
86.104.30.10             ether   00:11:2F:40:06:5A   CM                    eth1
86.104.30.8              ether   00:0A:E6:B7:CD:C9   CM                    eth1
86.104.30.5              ether   00:02:44:6D:89:54   CM                    eth1
86.104.31.74             ether   00:40:F4:B3:8C:96   C                     eth1
86.104.30.4              ether   00:03:0D:0D:FC:34   CM                    eth1
86.104.30.6              ether   08:00:46:E8:E8:B7   CM                    eth1
86.104.31.77             ether   00:30:05:29:60:9A   C                     eth1
86.104.30.9              ether   00:0B:CD:AB:88:B8   CM                    eth1
86.104.30.3              ether   00:50:04:31:AA:FD   CM                    eth1
86.104.30.7              ether   00:06:1B:DF:71:0B   CM                    eth1
86.104.30.2              ether   00:04:61:56:2A:9A   CM                    eth1
```

Este demn de remarcat faptul că *IP*-urile pentru care *MAC*-ul se învață în mod dinamic sunt marcate cu `dynamic`, în timp ce cele statice sunt marcate cu `static`.

ARP static în Linux

În cazul sistemului de operare *Linux*, legarea unui *IP* de un *MAC* se realizează folosind comanda `arp`[3] în același mod[4]:

```
arp -s ip mac
```

De asemenea, pentru a evita o succesiune de comenzi de forma:

```
# arp -s ip1 mac1
# arp -s ip2 mac2
...
# arp -s ipk mack
```

vom putea folosi comanda sub forma

```
arp -f fișier
```

unde fișierul dat ca argument conține una sau mai multe linii de forma:

```
ip mac
```

În exemplul din casetă, ridicăm o interfață virtuală cu *IP* `86.104.30.254/255.255.255.0`, după care încărcăm tabela de *MAC*-uri cu *IP*-uri din clasa `86.104.30. 0/24`.

Este demn de remarcat faptul că, în momentul vizualizării tabelei folosind comanda `arp`, înregistrările statice sunt marcate cu `CM`, în timp ce cele dinamice sunt marcate cu `C`.

ARP static în Cisco IOS

Adăugarea unei înregistrări statice se realizează folosind comanda:

```
arp ip mac încapsulare [alias]
```

unde parametrul *încapsulare* poate fi `ARPA`[5] sau `SNAP`[6], iar opțiunea *alias* se folosește pentru *PROXY-ARP*.

[3] *aveți nevoie de drepturi administrative pentru a rula comanda*
[4] *ștergerea se realizează folosind* `arp -d ip`
[5] *uzual, pentru interfețele de tip Ethernet*
[6] *uzual, pentru interfețele de tip FDDI și token-ring*

Comanda iptables

```
[root@server ~]# iptables -h
iptables v1.3.5

Usage: iptables -[AD] chain rule-specification [options]
       iptables -[RI] chain rulenum rule-specification [options]
       iptables -D chain rulenum [options]
       iptables -[LFZ] [chain] [options]
       iptables -[NX] chain
       iptables -E old-chain-name new-chain-name
       iptables -P chain target [options]
       iptables -h (print this help information)

Commands:
Either long or short options are allowed.
  --append  -A chain                Append to chain
  --delete  -D chain                Delete matching rule from chain
  --delete  -D chain rulenum        Delete rule rulenum (1 = first) from chain
  --insert  -I chain [rulenum]      Insert in chain as rulenum (default 1=first)
  --replace -R chain rulenum        Replace rule rulenum (1 = first) in chain
  --list    -L [chain]              List the rules in a chain or all chains
  --flush   -F [chain]              Delete all rules in  chain or all chains
  --zero    -Z [chain               Zero counters in chain or all chains
  --new     -N chain                Create a new user-defined chain
  --delete-chain
    -X [chain]                      Delete a user-defined chain
  --policy  -P chain target         Change policy on chain to target
  --rename-chain
    -E old-chain new-chain          Change chain name, (moving any references)

Options:
  --proto          -p [!] proto  protocol: by number or name, eg. `tcp'
  --source         -s [!] address[/mask]
                                 source specification
  --destination    -d [!] address[/mask]
                                 destination specification
  --in-interface   -i [!] input name[+]
                                 network interface name ([+] for wildcard)
  --jump           -j target
                                 target for rule (may load target extension)
  --match          -m match
                                 extended match (may load extension)
  --numeric        -n            numeric output of addresses and ports
  --out-interface  -o [!] output name[+]
                                 network interface name ([+] for wildcard)
  --table          -t table      table to manipulate (default: `filter')
  --verbose        -v            verbose mode
  --line-numbers                 print line numbers when listing
  --exact          -x            expand numbers (display exact values)
[!] --fragment     -f            match second or further fragments only
  --modprobe=<command>           try to insert modules using this command
  --set-counters PKTS BYTES      set the counter during insert/append
[!] --version      -V            print package version.
```

Ştergerea unei înregistrări statice se realizează folosind aceeaşi comandă, cu prefixul NO, conform uzanţelor *IOS*:

no arp *ip mac încapsulare* [*alias*]

Astfel, pentru echipamentul a cărui adresă *MAC* este 08:00:09:00:18:34, şi căruia i se atribuie *IP*-ul 192.168.7.19 comanda utilizată pentru adăugarea unei înregistrări statice este:

```
arp 192.168.7.19 0800.0900.1834 arpa
```

Blocare MAC din firewall

Următoarea idee de protecţie constă în blocarea accesului la nivel de *firewall*. Toate *firewall*-urile care operează la

nivel de *IP* trebuie să aibă și un mecanism de protecție bazat pe *MAC*.

Astfel, combinând cele două atribute, putem obține un bun *firewall*.

Blocare MAC din iptables sub Linux

Uzual, *firewall*-urile sub *Linux* sunt implementate folosind comanda `iptables`, a cărei sintaxă este prezentată în caseta de pe pagina anterioară[7].

Astfel implementarea firewall-ului va trebui sa contina in mod explicit secvente de comenzi de forma:

```
iptables -A chain -s ip_client -m mac
            --mac-source mac_client -j ACCEPT
```

Astfel, într-un *firewall* real ar putea să apară următoarele comenzi[8]:

```
...
iptables -A $LAN_CHAIN -s 172.1.1.102
     -m mac --mac-source 00:0D:61:9E:15:70
                                  -j ACCEPT
iptables -A $LAN_CHAIN -s 172.1.1.103
     -m mac --mac-source 00:0D:61:9E:15:1B
                                  -j ACCEPT
iptables -A $LAN_CHAIN -s 172.1.1.104
     -m mac --mac-source 00:0D:61:9E:11:76
                                  -j ACCEPT
...
```

Ocolirea protecției la nivel de MAC

Evident, protectia la nivel de *MAC* într-o retea nu este mecanismul infailibil pentru securizarea unei rețele.

Intrușii port să își asume destul de ușor roluri în numele altora prin simpla asumarea a *IP*-ului și a adresei *MAC* a țintei.

Cum putem vedea asocierile *MAC-IP*? Foarte simplu: prin interogarea tabelei de *MAC*-uri!

Cum putem vedea dacă este cineva *offline* pentru a ne putea asuma *IP*-ul și *MAC*-ul? Foarte simplu: folosind comanda `arping`.

Cum putem să ne asumam *IP*-ul și *MAC*-ul? Cu câteva drepturi administrative, putem realiza și acest lucru.

În episodul următor vom prezenta mai multe mecanisme de creștere a nivelului de securizare a rețelei și vom vedea care sunt prețurile plătite pentru fiecare dintre soluțiile descrise.

Probleme

Prezentăm în continuare enunțurile câtorva probleme (împreună cu soluțiile aferente) ale căror rezolvări se bazează, în principal, pe informațiile prezentate în cadrul acestui episod al serialului.

[7] *comandă rulată pe un sistem cu Fedora Core 5 cu actualizările la zi*

[8] *o secvență de comenzi* `iptables`

Problema #21

De ce într-o rețea *PAN*[9] sub *BlueTooth* pot fi cel mult opt membri?

Rezolvare

MAC-ul sub *BlueTooth* este pe trei biți.

Problema #22

ARP-ul static, ca metodă de securitate, oprește pachetele să iasă sau să intre dintr-un/într-un *router*?

Rezolvare

Pachetele care au ca destinație un anumit *IP* sunt atribuate pe câmpurile corespunzătoare la ieșirea din *router* cu *MAC*-ul destinatarului pe c\mpul corespunzător.

Dacă sistemul de *switch*-uri este bazat pe *switch*-uri *Layer 2* și fără niciun fel de management[10], acest *MAC* nu este găsit niciunde, deci un pachet spre acest *IP*[11] va fi trimis mai departe pe toate porturile, deci existând posibilitatea ca destinatarul să îl primească, deși este un *fake*[12].

Soluția constă în utilizarea protecției pe baza unui *MAC* valid în rețea[13], de încredere[14].

Problema #23

ARP-ul static pus pe un *router* va face ca pachetele care trebuie *route*-ate la un anumit *IP* cu *MAC fake* să nu ajungă? Dar un pachet de la un astfel de *IP*?

Rezolvare

vezi problema precedentă

Problema #24

De multe ori, într-o rețea de cartier cu protecție la nivel de *MAC* (*ARP* static, *firewall* cu `iptables` etc.) se observă o încărcare excesivă a procesorului[15] în zona `system`.

Care este rezolvarea care nu necesită investiții suplimentare?

Rezolvare

O soluție ar fi renunțarea la *ARP* static și protejarea pe baza de *firewall* realizat în mod inteligent (lanțuri de reguli cât mai scurte).

Problema #25

Avem o rețea de cartier în care avem alocate 256 de *IP*-uri având un *firewall* cu 256 de reguli consecutive; în cazul unei repartiții uniforme, care este media numărului regulilor verificate[16]? Generalizați!

[9] *Personal Area Network*

[10] *switch-uri "chioare"*

[11] *acest MAC*

[12] *intrus*

[13] *pe care să îl poată învăța switch-urile*

[14] *de exemplu, MAC-ul de la un echipament de rețea din backbone*

[15] *verificați, de exemplu, folosind comanda* `top` *sau interogând* `loadavg` *din* `/proc`

[16] *din punct de vedere al MAC-ului*

Rezolvare

Numărul mediu al regulilor este $(1 + 2 + ... + 256) / 256 = (256 \cdot 257 / 2) / 256 = 257 / 2 = 128.5$.

În cazul în care avem N reguli consecutive, presupunând repartiția uniformă, media este $(1 + 2 + ... + N) / N = (N \cdot (N + 1) / 2) / N = (N + 1) / 2$.

Problema #26

Avem o rețea de cartier în care avem alocate 256 de *IP*-uri și un *firewall* cu 256 de reguli consecutive; în cazul unei repartiții uniforme media regulilor verificate este 128.5 (vezi problema precedentă).

Descrieți o modalitate de a micșora media regulilor verificate la 17.

Rezolvare

Împărțim clasa `/24` în 16 clase `/28`. În 16 reguli consecutive, putem lua decizia de salt într-un lanț de reguli, lanț care va avea din nou 16 reguli consecutive.

Astfel cu o medie de 17 / 2 reguli verificate ajungem în lanțul de la al doilea doilea nivel, care este și el parcurs cu o medie de 17 / 2 reguli verificate.

Ca urmare, numărul total mediu de reguli verificate este 17 / 2 + 17 / 2 = 17.

Problema #27

Având o clasa de N *IP*-uri, care este modalitatea optimă[17] de organizare a regulilor într-un *firewall*, presupunând că nu avem restricție la numărul regulilor și lanțurilor.

În acest caz, care este numărul maxim de reguli care vor fi parcurse la procesarea unui pachet? Dar numărul minim?

Rezolvare

Organizăm lanțurile în mod arborescent, astfel încat un `/24` pe baza unei reguli se va continua cu unul dintre cele doua `/25`-uri.

În acest caz, cele N *IP*-uri se vor organiza într-un arbore binar cu înălțimea[18] egală cu partea întreagă superioară a valorii $\log_2 N$. De exemplu, pentru un `/24` vom avea o înălțime de `16`.

Numărul regulilor procesate va fi întotdeauna același în cazul în care N este o putere a lui 2.

Problema #28

Care este dezavantajul metodei alese pentru rezolvarea problemei anterioare? Care ar fi compromisul?

Rezolvare

Dezavantajul este dat în special de limitarea numărului maxim al regulilor și al lanțuri, respectiv de timpul care se pierde prin abandonarea lanțului curent și saltul într-un lanț nou.

O altă problema care apare este dată de generarea acestor *firewall*-uri:

- de regulă, administratorii de rețea și de administratorii de sistem nu sunt cei mai buni programatori;
- numărul mare al regulilor poate îngreuna o repornire a *firewall*-ului.

Soluțiile aplicate în general sunt soluții mixte bazate pe o ierarhie pe trei niveluri:

- la primul nivel se selectează `/24`-ul corespunzăto *IP*-ului;
- la al doilea nivel se realizează o selecție a `/28`-ului;
- la al treilea nivel se realizează verificarea efectivă.

Problema #29

De ce există echipamente *wireless*[19] care nu pot funcționa fără a avea "în spate" o placă de rețea legată pe *Ethernet*?

Rezolvare

Respectivele echipamente nu posedă *MAC* propriu pe interfața *bridge* construita peste *RF* și *Ethernet*.

Problema #30

De ce există unele echipamente de tip *cable modem* sau *ADSL modem* care ofera trei *MAC*-uri distincte pe interfețele *WAN*, *Ethernet* și *USB*?

Rezolvare

Respectivele echipamente pot functiona, eventual[20], și în alt regim decât cel de *bridge* (chiar dacă la echipamentele de tip *cable modem* sunt, uneori, greu de modificat configurările implicite).

Bibliografie

1. Tiberiu Socaciu, *Lecții de rețele*, vol. 1, Editura InfoData, Cluj-Napoca, 2007;
2. Tiberiu Socaciu, *MAC și ARP (I)*, *GInfo* 17/2, februarie 2007, Editura Agora Media, Târgu Mureș;
3. Tiberiu Socaciu, *MAC și ARP (II)*, *GInfo* 17/3, martie 2007, Editura Agora Media, Târgu Mureș;
4. Tiberiu Socaciu, *VLAN, TRUNK și STP*, *GInfo* 17/1, ianuarie 2007, Editura Agora Media, Târgu Mureș;
5. Tiberiu Socaciu, *Tuneluri și bridge-uri*, *GInfo* 16/8, decembrie 2006, Editura Agora Media, Târgu Mureș;
6. Tiberiu Socaciu, *Limitări de viteză în rețele*, *GInfo* 16/6, octombrie 2006, Editura Agora Media, Târgu Mureș;
7. Tiberiu Socaciu, *Informatica privită... altfel sau 16 chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft, Bacău, 2006;
8. Tiberiu Socaciu, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura InfoData, Cluj-Napoca, 2006;
9. Tiberiu Socaciu, *O altfel de ... Informatică sau diverse chestiuni de matematică, backtracking, clustere de calculatoare, Linux, inteligență artificială și networking*, Editura InfoData, Cluj-Napoca, 2007.

[17] *din punct de vedere al numărului regulilor*

[18] *înălțimea va indica și numărul regulilor care vor fi procesate*

[19] *de exemplu, unele echipamente în 2,4GHz în tehnologie FHSS*

[20] *uneori, rescriind flash-ulM alteori, folosind funcțiile explicite oferite de soft-ul de bază al echipamentului*

INTELIGENȚA ARTIFICIALĂ între ADEVĂR și MIT

Ionuț Chirilă

Continuând expunerea precedentă, interesant ar fi să trecem în revistă modurile principale de reprezentare a unui fapt de inteligență artificială. Cel mai întâlnit pare a fi, de departe, recunoașterea unui tipar sau "pattern recognition".

În aceste programe, motorul compară ceea ce găsește cu modelele aflate în baza sa de cunoștințe inițială. Un exemplu concret este constituit de textul în limbaj natural. Aici avem de găsit potriviri de structură cu un șablon dat. Cele mai simple programe caută asemănările de formă (structură), în timp ce cele mai complexe vor căuta combinațiile semantice care se potrivesc.

În expunerea trecută am văzut și un exemplu de inferență. Concluziile sunt construite pe baza premiselor. Sunt posibile și cazurile în care o premisă va fi introdusă automat (default), existând cazuri în care poate fi retrasă în cazul descoperirii unei fapte contrare celei care susține premisa introdusă. Aici se pot încadra cu ușurință majoritatea problemelor de clasificare și selecție.

Un exemplu interesant îl constituie "învățarea din experiență"; programul învață doar care fapte sau formalisme ale regulilor pot fi reprezentate sub formă de informații.

Categoria cea mai mult inspirată din pragmatismul lumii reale este constituită de programele de *planning*. Acestea încep cu faptele care definesc o situație particulară din care caută să găsească scopul final al problemei propuse. Se generează o strategie de atingere a scopului.

Faptele trebuie reprezentate într-un anumit fel. De regulă, aici intervine modelul logicii matematice. După cum observăm, este vorba despre o reprezentare a formei și nu a fondului reprezentat de fapte.

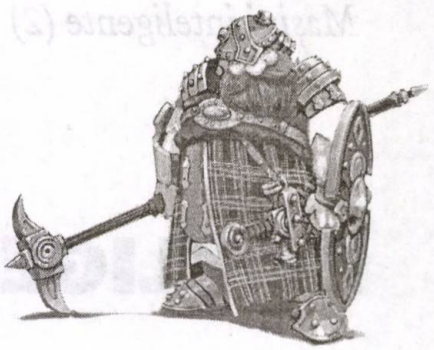
De departe însă, modul cel mai cuprinzător și atractiv este modul de reprezentare al ingineriei genetice, care se poate afla în strânsă legătură cu modelul rețelelor neuronale.

Scopul final al acestor moduri de prezentare este de a "subjuga" definitiv mașina și a asigura un parteneriat durabil în lupta cu informația în starea ei pură. Între aplicațiile răsunătoare putem aminti: mașina poate bate la șah orice mare maestru internațional, poate recunoaște și reproduce vocea umană cu rezultate apreciabile. De asemenea, mașina poate înțelege limbajul natural cu tot ce presupune acesta. Adăugăm aici sistemele expert care permit interacțiunea dintre un inginer uman și mașină într-un anumit domeniu cu scopul de a duce la bun sfârșit o anumită sarcină.

Am văzut la nivel de concept ce înseamnă inteligența artificială sau o mașină inteligentă. Dar cum se reprezintă efectiv aceasta? De ce nu se poate folosi un limbaj de programare clasic? Aici, avantajul principal este cel al folosirii unui limbaj cât mai apropiat de cel natural.

La fel cum mintea umană are la bază neuronul și informația păstrată de acesta, se poate considera o paralelă în care *IA* are în centru faptul - ordonat sau neordonat. Alt avantaj în limbajele de *IA* este ușurința construirii de modele și asociatii de *IA*. Programarea clasică se oprește la nivelul structurii în sine. Limbajele de *IA* (*CLIPS*, *Jess* etc.) modelează într-un mod facil și imaginea structurii, adică informația. Același lucru al ușurinței în exploatare se poate menționa și în ceea ce privește bazele de cunoștințe în raport cu bazele de date tradiționale. Ultimele au restricția omogenității datelor, în timp ce primele nu au această limitare.

Dwarfi în Ashan



HEROES of MIGHT and MAGIC V: HAMMERS of FATE

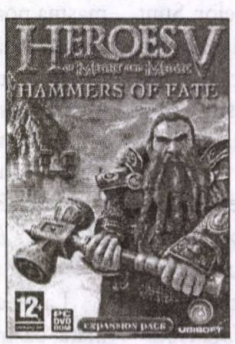
Mihai Scorțaru

La șase luni după lansarea jocului Heroes V, a apărut pe piață primul add-on al acestuia. Au apărut dwarfii, au revenit caravanele, avem trei noi campanii, noi unități neutre etc.

Dwarfii

Cea mai importantă noutate o constituie apariția celei de-a șaptea rase din *Ashan*: dwarfii. Aceștia îl venerează pe *Arkath*, dragonul focului, trăiesc în *Grimheim*, *Regatul de sub munte*, capitala aflându-se la *Tor Myrdal*.

Dwarfii sunt războinici feroce, fierari renumiți, buni marinari și maeștrii ai magiei runice. Ei sunt suspicioși cu cei din exterior, primii gardieni ai demonilor și rivali ai elfilor întunecați.



Fiecare oraș al dwarfilor are propriul rege și un *Sfat* alcătuit din reprezentanții celor mai bogate familii. *Regele de sub munte* este regele tuturor regilor și el este cel care stabilește linia politică generală și cel care poate declara războiul. Titlul de rege nu este transmis de la o generație la alta, deși cel care îl deține, îl păstrează pe tot timpul vieții.

După moartea unui rege, este convocat un consiliu care alege succesorul. În astfel de momente, tunelurile în care trăiesc dwarfii sunt în sărbătoare, dar apar și intrigile, deoarece diferiți candidați încearcă să îi convingă pe cei care votează și să își discrediteze sau să își elimine rivalii.

Cultura dwarfilor este dominată de bărbați deși, contrar zvonurilor, femeile nu sunt atât de rare pe cât s-ar putea crede și nici nu au barbă.

Structura socială se bazează pe trei tipuri de relații: familie, adopție și luptă. Există o rețea complicată de ucenicie care servește la antrenarea tinerilor dwarfi, dar și la cimentarea relațiilor dintre clanuri. Aceste legături sunt foarte prețioase; un singur ucenic nevrednic poate duce la distrugerea unei alianțe vechi de secole.

Deși există numeroase tipuri de legături care țin regatul unit (de rasă, de familie sau de oraș), cea mai importantă dintre ele este cea de clan. Cele șase clanuri ale dwarfilor sunt:

- *Deepflame* - preoți ai runelor, conducătorii religioși ai dwarfilor;
- *Grimsteel* - un clan feroce care se remarcă prin faimoșii săi războinici;
- *Stonefist* - clanul responsabil cu mineritul și cu clădirea marilor orașe subterane;
- *Stronghammer* - artizani și fierari, creatorii faimoaselor arme sau alte obiecte ale dwarfilor;
- *Hearthguard* - un clan al magistraților și al administratorilor; întotdeauna lenți și serioși.
- *Winterwind* - maeștrii animalelor, obișnuiți să trăiască afară și prietenoși cu alte rase.

Un clan nu este același lucru cu o familie. Familile reprezintă diferite linii ereditare în cadrul aceluiași clan. Fiecare clan are propriul nume, propria istorie a bătăliilor, propriile obiceiuri și așa mai departe. Până la urmă, fiecare dwarf se întoarce la clanul său.

Deși un oraș poate să adăpostească dwarfi din diferite clanuri, în momente importante, cum ar fi alegerea unui nou

rege, dwarfii apără înainte de orice interesele clanului din care fac parte.

Războinicii dwarfi reprezintă una dintre cele mai temute forțe din întreaga lume. Cu o ferocitate de neegalat și o disciplină de fier, armata poate fi chemată la luptă foarte repede.

În luptă, grupele de războinici din diferitele orașe se întrec între ele numărând inamicii uciși, drapelele capturate și diverse alte elemente de acest tip. Războiul este un joc și aceasta este o modalitate de a ține scorul.

Pe de altă parte, dwarfii nu se predau niciodată; luptă până la moarte chiar dacă șansele împotriva lor sunt copleșitoare. Cea mai bună moarte pentru un dwarf este în luptă, cu șanse imposibil de mici de a câștiga și cu un martor care, într-o zi, va compune un cântec despre faptele sale.

Deși preferă să lupte sub pământ, dwarfii ies uneori la suprafață. În tunele, unde bandele mari de războinici nu sunt foarte utile, dwarfii construiesc fortificații și capcane complicate.

Dwarfii care explorează tuneluri necunoscute sunt numiți delveri și ei reprezintă, de obicei, prima linie defensivă în fața incursiunilor elfilor întunecați, ale necromanților sau ale altor inamici.

Istoria

Dwarfii erau împărțiți în mai multe clanuri împrăștiate, dar s-au unit într-o grandioasă națiune, capabilă să se apere împotriva tuturor inamicilor. Copii ai dragonului focului, ei îl venerează pe acesta în locuri aflate la mare adâncime, acolo unde singura sursă de lumină este magma. În schimb, ei au învățat secretele fierăriei și ale inimii.

La început, celelalte rase nici nu știau de existența dwarfilor. Doar în timpul *Războaielor Focului* (dintre anii 28 și 41 ai *Celui de-al Șaptelea Dragon*), când războinicii dwarfi s-au alăturat celorlalți pentru a rezista în fața demonilor, au intrat ei în contact cu celelalte rase.

După încheierea luptelor, dwarfii s-au retras în casele lor de sub munți și nu am mai prea avut de-a face cu celelalte rase timp de secole. În acest timp, tuneluri erau săpate lent între orașe dar, până la urmă, toate orașele au fost legate printr-o rețea aflată mult sub suprafața pământului.

Toate acestea s-au schimbat când elfii întunecați s-au stabilit la granițele dwarfilor. Relațiile care se deteriorau rapid au dus la *Războiul de sub Munte* în care dwarfii, conduși de *Hathor Deepstrider Orlandsson*, i-au zdrobit pe elfii întunecați și i-au pus pe fugă.

Locurile în care trăiau elfii întunecați au fost distruse și dwarfii și-au considerat granițele securizate.

Astăzi, dwarfii își privesc cu vigilență granițele și nimeni nu poate păși nevăzut pe pământurile lor. Se împrietenesc cu puțini și își extind cu atenție regatul.

Majoritatea istoriei dwarfilor este ascunsă de restul lumii. Dwarfii au lungi legende, epopee și ode care descriu faptele și războaiele lor, dar care nu sunt împărtășite celor din afară.

Lucrurile pe care le-au văzut și luptele pe care le-au purtat în interiorul granițelor lor rămân un mister, cel puțin deocamdată.

Unitățile dwarfilor

Defender
Formează temelia oricărei armate a dwarfilor și sunt cele mai numeroase trupe. Sprijiniți de puternice armuri, pot să facă față atacurilor unor inamici mult mai puternici.

Shieldguard
Sunt războinici călițiți în luptă, a căror misiune este de a proteja celelalte unități. Folsindu-și magnificele scuturi, pot crea un zid de nestrăpuns care duce la scăderea daunelor provocate de inamici. Cu cât este mai mare distanța parcursă de inamic pentru atac, cu atât lovitura va avea o putere mai mică.

Spearwielder
Reprezintă unul dintre atuurile oricărei armate a dwarfilor. Șocul atacului lor și talentul în mânuirea tăioaselor lance le oferă șansa de a scădea viteza cu care reacționează inamicii, precum și inițiativa acestora.

Skirmisher
Sunt cei care au reușit să stăpânească arta de a-și arunca armele spre adversari. Pot ataca de la distanță, având același efect asupra vitezei și inițiativei adversarilor. Sunt la fel de eficienți și în lupta corp la corp.

Bear Rider
Nicio armată nu poate face mare lucru fără cavalerie, al cărei scop este acela de a ajunge și a angaja rapid în luptă inamicul. Dwarfii au domesticit ferocii urși în acest scop, deoarece forța și rezistența acestora compensează viteza relativ scăzută cu care se deplasează.

Blackbear Rider
Sunt trupele călare de elită ale dwarfilor. Bestiile folosite poartă armuri grele și sunt antrenate cu rigurozitate pentru a li se îmbunătăți vitalitatea. În plus, ursii negri pot participa activ la luptă și își pot lovi adversarii cu labele, ceea ce duce la scăderea inițiativei acestora.

Brawler
Maeștrii ai luptei corp la corp, ei nici măcar nu folosesc arme. Se bazează pe utilizarea unor mănuși de fier și pe puterea propriilor mâini și picioare, care sunt suficiente pentru a provoca daune importante. Sunt atât de dedicați artelor marțiale pe care le folosesc, încât sunt imuni la vrăjile care afectează mintea.

Berseker
Mănușile sunt dotate cu lame. În timpul bătăliei pot renunța complet la tacticile defensive, ceea ce duce la creșterea impresionantă a forței de atac. Totuși, când folosesc această strategie, nu își dau seama cine se află în jurul lor și atacă cea mai apropiată unitate (chiar dacă este aliată). Datorită surprizei, unitatea atacată nu poate răspunde la atac.

Rune Priest
Sunt o castă specială în rândul dwarfilor care, recrutați din copilărie datorită ochilor aurii, semnul zeului lor Arkath, devin vrăjitori. Își lovesc adversarii de la distanță și sunt rezistenți la atacurile magice ale acestora. Pot să își afecteze inamicii cu semnul focului, ceea ce face ca aceștia să sufere mai mult de pe urma magiilor de foc.

Rune Patriarch
Preoții înzestrați își obțin consacrarea în cele mai secrete săli din Deepflame. Atacurile lor își ating inamicii și apoi afectează toate celalalte unități din jurul țintelor.

Unitățile dwarfilor

Thane
Nu sunt doar războinici redutabili, dar au și talente magice. Se pot teleporta în interiorul liniilor inamice și își pot măcelări adversarii folosind forța brută a securilor. În plus, securile lor provoacă fulgere care atacă inamicii din apropiere. De asemenea, ei sunt imuni la fulgere.

Warlord
Și-au dedicat viețile studiului puterii furtunilor din munți. În timpul luptei, o singură dată, pot să trimită un fulger puternic care să lovească o țintă, împreună cu toate unitățile din jurul acesteia.

Fire Dragon
Sunt copiii lui Arkath, dragonul focului și zeul dwarfilor. La fel ca dragonii negri, rudele lor îndepărtate, aceste creaturi maiestuoase își pot ataca inamicii folosindu-se de răsuflarea incandescentă și răspund la atacurile inamice cu jerbe de foc, asemeni păsărilor phoenix. Evident, sunt complet imuni la foc.

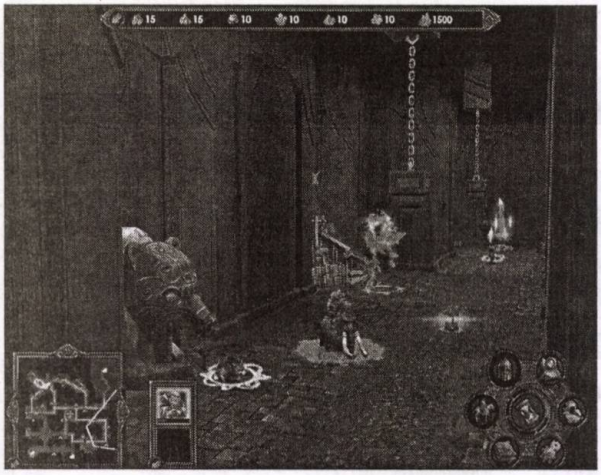
Magma Dragon
Par a fi cu adevărat formați din piatră arzândă. Platoșele de piatră străbătute de nervuri incandescente fac ca acești giganți arzători să fi protejați de o armură aproape de nepătruns.

Creaturile

La fel ca și pentru celelalte șase rase, există șapte tipuri de creaturi (șapte niveluri), pentru fiecare dintre acestea existând un upgrade. În casetă sunt prezentate imaginile unităților, împreună cu o scurtă prezentare a fiecăreia.

Caracteristicile unităților sunt următoarele:

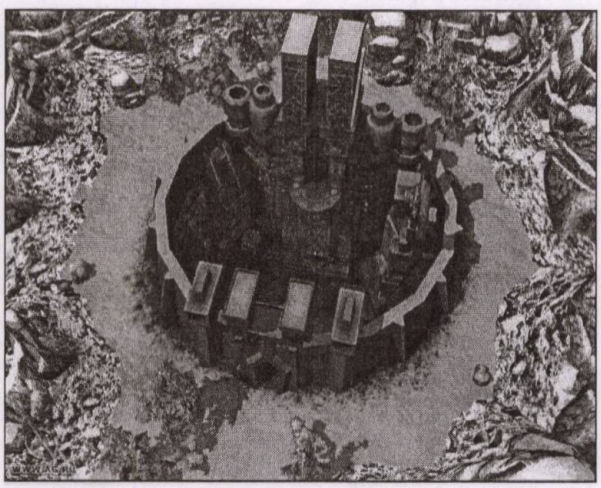
- *Defender:*
  - *Atac*: 1;
  - *Apărare*: 4;
  - *Interval pentru daunele produse*: 1 - 1;
  - *Inițiativă*: 9;
  - *Viteză*: 3;
  - *Vitalitate*: 7;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 24 aur;
  - *Rata de creștere săptămânală*: 18;
  - *Experiență obținută de adversar*: 7;
  - *Evaluarea puterii*: 70;
  - *Abilități speciale:*
    - ◊ *Enraged:* atacul crește în timpul luptei în cazul în care o unitate aliată piere (cu excepția unităților readuse la viață sau a unităților invocate prin intermediul magiei);
    - ◊ *Armoured:* unitatea este rezistentă la toate vrăjile care scad apărarea;
    - ◊ *Large Shield:* unitatea suferă doar 50% din daunele provocate în mod normal de atacurile la distanță (cu excepția celor magice);
- *ShieldGuard:*
  - *Atac*: 1;
  - *Apărare*: 5;
  - *Interval pentru daunele produse*: 1 - 2;
  - *Inițiativă*: 9;
  - *Viteză*: 3;
  - *Vitalitate*: 12;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 40 aur;
  - *Rata de creștere săptămânală*: 18;
  - *Experiență obținută de adversar*: 12;
  - *Evaluarea puterii*: 115;
  - *Abilități speciale:*
    - ◊ *Enraged:* atacul crește în timpul luptei în cazul în care o unitate aliată piere (cu excepția unităților readuse la viață sau a unităților invocate prin intermediul magiei);
    - ◊ *Armoured:* unitatea este rezistentă la toate vrăjile care scad apărarea;
    - ◊ *Large Shield:* unitatea suferă doar 50% din daunele provocate în mod normal de atacurile la distanță (cu excepția celor magice);
    - ◊ *Shield Wall*: daunele suferite de această unitate în urma atacurilor inamice sunt reduse cu 10% pentru fiecare celulă parcursă de inamic pentru a ataca (reducerea maximă este de 90%);

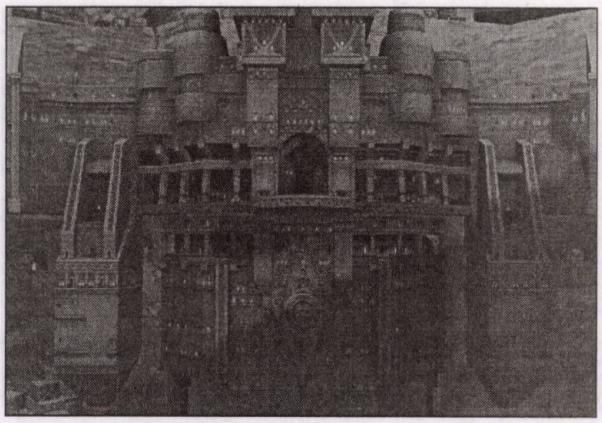


- *Spearwielder:*
  - *Atac*: 4;
  - *Apărare*: 4;
  - *Interval pentru daunele produse*: 1 - 2;
  - *Inițiativă*: 9;
  - *Viteză*: 4;
  - *Vitalitate*: 10;
  - *Mana*: 0;
  - *Lovituri la distanță*: 2;
  - *Cost*: 45 aur;
  - *Rata de creștere săptămânală*: 14;
  - *Experiență obținută de adversar*: 11;
  - *Evaluarea puterii*: 110;
  - *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *Crippling Wound:* în urma unui atac, există șansa ca viteza și inițiativa țintei să se reducă (viteza scade cu 50%, iar inițiativa cu 30%); efectul durează timp de două ture;
- *Skirmisher:*
  - *Atac*: 4;
  - *Apărare*: 4;
  - *Interval pentru daunele produse*: 2 - 3;
  - *Inițiativă*: 9;
  - *Viteză*: 4;
  - *Vitalitate*: 12;
  - *Mana*: 0;
  - *Lovituri la distanță*: 4;
  - *Cost*: 65 aur;
  - *Rata de creștere săptămânală*: 14;
  - *Experiență obținută de adversar*: 17;
  - *Evaluarea puterii*: 156;
  - *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *No Melee Penalty*: unitatea nu este penalizată în timpul atacului corp la corp;
    - ◊ *Crippling Wound:* în urma unui atac, există șansa ca viteza și inițiativa țintei să se reducă (viteza scade cu 50%, iar inițiativa cu 30%); efectul durează timp de două ture;
- *Bear Rider:*
  - *Atac*: 5;
  - *Apărare*: 10;
  - *Interval pentru daunele produse*: 4 - 5;
  - *Inițiativă*: 10;
  - *Viteză*: 6;
  - *Vitalitate*: 25;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 130 aur;
  - *Rata de creștere săptămânală*: 7;
  - *Experiență obținută de adversar*: 24;
  - *Evaluarea puterii*: 304;
  - *Abilități speciale:*
    - ◊ *Enraged:* atacul crește în timpul luptei în cazul în care o unitate aliată piere (cu excepția unităților readuse la viață sau a unităților invocate prin intermediul magiei);
    - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
- *Blackbear Rider:*
  - *Atac*: 6;
  - *Apărare*: 14;
  - *Interval pentru daunele produse*: 5 - 6;
  - *Inițiativă*: 11;
  - *Viteză*: 7;
  - *Vitalitate*: 30;
  - *Mana*: 0;

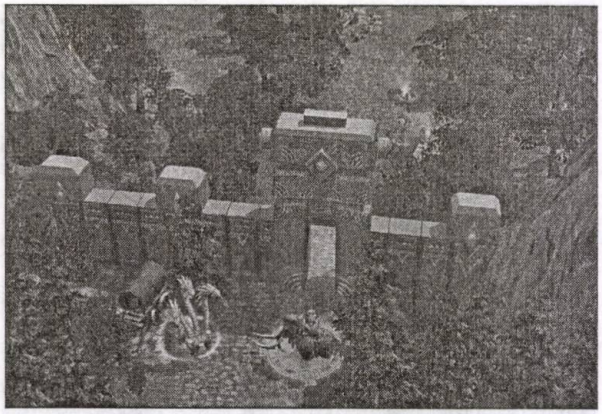
- *Lovituri la distanță*: 0;
- *Cost*: 185 aur;
- *Rata de creștere săptămânală*: 7;
- *Experiență obținută de adversar*: 36;
- *Evaluarea puterii*: 419;
- *Abilități speciale:*
  - ◊ *Enraged:* atacul crește în timpul luptei în cazul în care o unitate aliată piere (cu excepția unităților readuse la viață sau a unităților invocate prin intermediul magiei);
  - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
  - ◊ *Armoured:* unitatea este rezistentă la toate vrăjile care scad apărarea;
  - ◊ *Paw Strike:* unitatea are șansa de a-și împinge ținta cu o celulă în spate, anulând toată inițiativa acumulată de aceasta; șansa depinde de numărul celulelor parcurse pentru efectuarea atacului; un *treant* atacat (care își folosește abilitatea *entangle*) va suferi penalizarea de inițiativă, dar nu se va deplasa și unitățile prinse de acesta nu vor fi eliberate;

- *Brawler:*
  - *Atac*: 6;
  - *Apărare*: 6;
  - *Interval pentru daunele produse*: 2 - 6;
  - *Inițiativă*: 12;
  - *Viteză*: 5;
  - *Vitalitate*: 20;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 160 aur;
  - *Rata de creștere săptămânală*: 6;
  - *Experiență obținută de adversar*: 27;
  - *Evaluarea puterii*: 317;
  - *Abilități speciale:*
    - ◊ *Immunity to Mind Control:* unitatea este imună la vrăjile care afectează mintea;
- *Berserker:*
  - *Atac*: 7;
  - *Apărare*: 7;



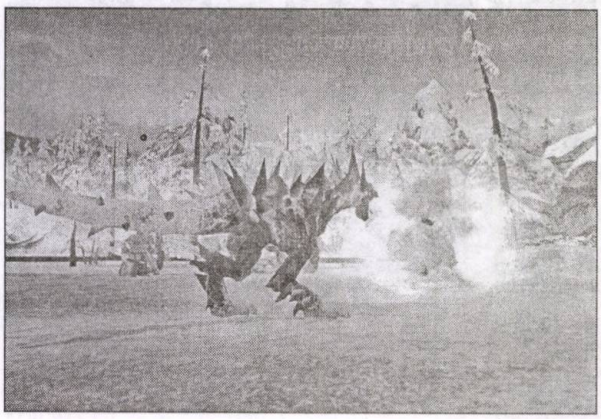


  - *Interval pentru daunele produse*: 3 - 8;
  - *Inițiativă*: 12;
  - *Viteză*: 5;
  - *Vitalitate*: 25;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 215 aur;
  - *Rata de creștere săptămânală*: 6;
  - *Experiență obținută de adversar*: 42;
  - *Evaluarea puterii*: 440;
  - *Abilități speciale:*
    - ◊ *Immunity to Mind Control:* unitatea este imună la vrăjile care afectează mintea;
    - ◊ *Berserker Rage:* în momentul utilizării acestei abilități, apărarea creaturii scade la 0 și atacul crește cu valoarea cu care a scăzut apărarea; este atacat cel mai apropiat inamic și acesta nu răspunde la atac; dacă în raza de acțiune nu se află niciun inamic, atunci este atacată cea mai apropiată unitate aliată;
- *Rune Priest:*
  - *Atac*: 10;
  - *Apărare*: 6;
  - *Interval pentru daunele produse*: 12 - 15;
  - *Inițiativă*: 8;
  - *Viteză*: 3;
  - *Vitalitate*: 60;
  - *Mana*: 15;
  - *Lovituri la distanță*: 5;
  - *Cost*: 470 aur;
  - *Rata de creștere săptămânală*: 3;
  - *Experiență obținută de adversar*: 59;
  - *Evaluarea puterii*: 819;
  - *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *Caster:* unitatea poate folosi anumite vrăji, în mod asemănător eroilor; cantitatea de mana este limitată,



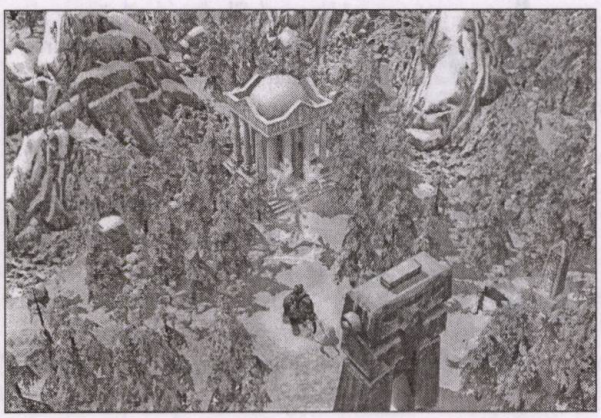
dar este întotdeauna la nivelul maxim la începutul luptei; puterea magică depinde de numărul creaturilor;
◊ *Magic-proof 50%*: daunele produse de atacurile magice sunt reduse cu 50%;
◊ *Mark Of Fire:* în urma unui atac, există șansa ca ținta să fie marcată cu semnul focului ceea ce face ca, timp de două runde, aceasta să sufere de două ori mai multe daune în urma atacurilor cu foc;
◊ *Cartea cu vrăji:*
→ *Deflect Missile* (6 mana): unitatea aliată selectată va suferi mai puține daune în urma atacurilor la distanță;

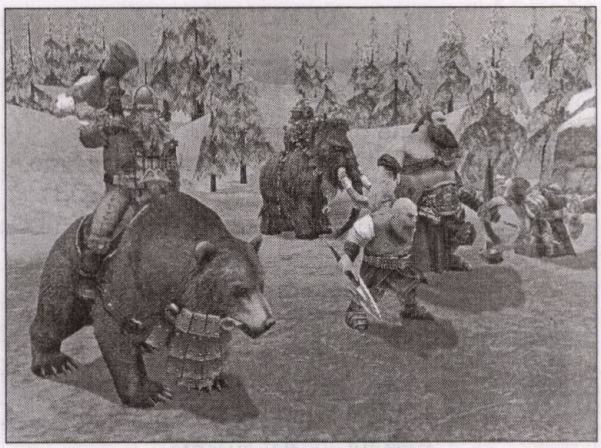
- *Rune Patriarch:*
  - ♦ *Atac*: 10;
  - ♦ *Apărare*: 9;
  - ♦ *Interval pentru daunele produse*: 14 - 18;
  - ♦ *Inițiativă*: 9;
  - ♦ *Viteză*: 3;
  - ♦ *Vitalitate*: 70;
  - ♦ *Mana*: 25;
  - ♦ *Lovituri la distanță*: 5;
  - ♦ *Cost*: 670 aur;
  - ♦ *Rata de creștere săptămânală*: 3;
  - ♦ *Experiență obținută de adversar*: 100;
  - ♦ *Evaluarea puterii*: 1129;
  - ♦ *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *Caster:* unitatea poate folosi anumite vrăji, în mod asemănător eroilor; cantitatea de mana este limitată, dar este întotdeauna la nivelul maxim la începutul luptei; puterea magică depinde de numărul creaturilor;
    - ◊ *Magic-proof 50%*: daunele produse de atacurile magice sunt reduse cu 50%;
    - ◊ *Mark Of Fire:* în urma unui atac, există șansa ca ținta să fie marcată cu semnul focului ceea ce face ca, timp de două runde, aceasta să sufere de două ori mai multe daune în urma atacurilor cu foc;
    - ◊ *Crossfire:* zona de atac a acestei unități are forma unei cruci;
    - ◊ *Cartea cu vrăji:*
      - → *Deflect Missile* (6 mana): unitatea aliată selectată va suferi mai puține daune în urma atacurilor la distanță;
      - → *Firewall* (16 mana): creează un zid de foc în zona selectată; timp de câteva tururi creaturile din zona afectată suferă daune (magie de foc);
- *Thane:*
  - ♦ *Atac*: 15;
  - ♦ *Apărare*: 25;
  - ♦ *Interval pentru daunele produse*: 8 - 12;
  - ♦ *Inițiativă*: 11;
  - ♦ *Viteză*: 8;
  - ♦ *Vitalitate*: 100;
  - ♦ *Mana*: 0;
  - ♦ *Lovituri la distanță*: 0;
  - ♦ *Cost*: 1300 aur;
  - ♦ *Rata de creștere săptămânală*: 2;
  - ♦ *Experiență obținută de adversar*: 131;
  - ♦ *Evaluarea puterii*: 2133;
  - ♦ *Abilități speciale:*
    - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
    - ◊ *Teleport:* unitatea se poate teleporta pe câmpul de luptă, evitând obstacolele;
    - ◊ *Immunity to Lightning:* fulgerele nu pot afecta această unitate;
    - ◊ *Storm Strike:* pe lângă daunele obișnuite, atacul acestei unități provoacă fulgere care lovesc o altă țintă aflată lângă prima, continuându-se în acest mod până în momentul în care lanțul se rupe (nu mai există adversari lângă ultima unitate atinsă); forța fulgerului este egală cu cea a atacului inițial;
- *Warlord:*
  - ♦ *Atac*: 15;
  - ♦ *Apărare*: 25;
  - ♦ *Interval pentru daunele produse*: 9 - 14;
  - ♦ *Inițiativă*: 11;
  - ♦ *Viteză*: 8;



- *Vitalitate*: 120;
- *Mana*: 0;
- *Lovituri la distanță*: 0;
- *Cost*: 1700 aur;
- *Rata de creștere săptămânală*: 2;
- *Experiență obținută de adversar*: 162;
- *Evaluarea puterii*: 2505;
- *Abilități speciale:*
  - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
  - ◊ *Teleport:* unitatea se poate teleporta pe câmpul de luptă, evitând obstacolele;
  - ◊ *Immunity to Lightning:* fulgerele nu pot afecta această unitate;
  - ◊ *Storm Strike:* pe lângă daunele obișnuite, atacul acestei unități provoacă fulgere care lovesc o altă țintă aflată lângă prima, continuându-se în acest mod până în momentul în care lanțul se rupe (nu mai există adversari lângă ultima unitate atinsă); forța fulgerului este egală cu cea a atacului inițial;
  - ◊ *Storm Bolt:* în timpul bătăliei, o singură dată, unitatea poate ataca orice țintă folosindu-și pentru aceasta abilitatea *Storm Strike*;

- *Fire Dragon:*
  - *Atac*: 25;
  - *Apărare*: 35;
  - *Interval pentru daunele produse*: 40 - 50;
  - *Inițiativă*: 9;
  - *Viteză*: 5;
  - *Vitalitate*: 230;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 3500 aur, un cristal;
  - *Rata de creștere săptămânală*: 1;
  - *Experiență obținută de adversar*: 255;
  - *Evaluarea puterii*: 5022;
  - *Abilități speciale:*
    - ◊ *Fire Breath:* în timpul atacului, unitatea provoacă daune nu numai țintei, ci oricărei alte unități amplasate în spatele acesteia;
    - ◊ *Fire Shield:* în momentul în care unitatea este atacată în luptă corp la corp, scutul său de foc provoacă atacatorului daune, indiferent dacă acesta are sau nu abilitatea *counter-attack*; atacatorul suferă daune (magie de foc) egale cu 20% din daunele cauzate.
    - ◊ *Elemental:* această unitate nu este vie și nu poate fi afectată de otravă, orbire sau de magii care afectează mintea; de asemenea, nu poate fi vindecată sau readusă la viață;
    - ◊ *Immunity to Fire:* vrăjile de foc nu pot afecta această unitate;





- *Magma Dragon:*
  - *Atac*: 30;
  - *Apărare*: 40;
  - *Interval pentru daunele produse*: 40 - 50;
  - *Inițiativă*: 9;
  - *Viteză*: 5;
  - *Vitalitate*: 280;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 4500 aur, două cristale;
  - *Rata de creștere săptămânală*: 1;
  - *Experiență obținută de adversar*: 329;
  - *Evaluarea puterii*: 6100;
  - *Abilități speciale:*
    - ◊ *Fire Breath:* în timpul atacului, unitatea provoacă daune nu numai țintei, ci oricărei alte unități amplasate în spatele acesteia;
    - ◊ *Magma Shield:* în momentul în care unitatea este atacată în luptă corp la corp, scutul său de magmă provoacă atacatorului daune; atacatorul suferă daune (magie de foc) egale cu 40% din daunele cauzate;
    - ◊ *Elemental:* această unitate nu este vie și nu poate fi afectată de otravă, orbire sau de magii care afectează mintea; de asemenea, nu poate fi vindecată sau readusă la viață;
    - ◊ *Immunity to Fire:* vrăjile de foc nu pot afecta această unitate.

Magia runică

Caracteristica rasială a dwarfilor constă în posibilitatea de a utiliza magia runică. Pentru aceasta, în orașele lor pot fi construite altare runice care permit eroilor să învețe magii runice. La fel ca în cazul ghildelor magice obișnuite (care pot

Rune

Rune of Berserking (nivelul 1)
Cost: un minereu
Unitatea aleasă va efectua două atacuri corp la corp în loc de unul. Poate fi activată doar dacă cel puțin o creatură a pierit deja.

Rune of Charge (nivelul 1)
Cost: un lemn
Viteza unității alese va crește cu 100%. Efectul va dura până la sfârșitul rundei curente.

Rune of Exorcism (nivelul 2)
Cost: un sulf
Înlătură toate efectele negative care afectează unitatea aleasă (doar cele care pot fi înlăturate).

Rune of Magic Control (nivelul 2)
Cost: un giuvaier
Unitatea aleasă va avea șansa de a "fura" un efect pozitiv aleator care afectează unitatea atacată.

Rune of Elemental Imunity (nivelul 3)
Cost: un mercur
Unitatea aleasă va deveni imună, până la sfârșitul luptei, la două dintre cele patru elemente magice (foc, aer, apă și pământ); cele două elemente sunt alese aleator.

Rune of Etherealness (nivelul 3)
Cost: un cristal
Unitatea aleasă devine acorporală timp de o rundă.

Rune of Resurrection (nivelul 4)
Cost: un minereu, un giuvaier
40% dintre creaturile ucise sunt readuse la viață.

Rune of Thunderclap (nivelul 4)
Cost: un lemn, un mercur
Există șansa ca unitatea atacată să își piardă inițiativa.

Rune of Battle Rage (nivelul 5)
Cost: un minereu, un cristal
Sunt atacate toate unitățile inamice aflate în celule adiacente unității alese, fără ca acestea să riposteze.

Rune of Dragonform (nivelul 5)
Cost: un lemn, un sulf
Atacul și apărarea unității alese cresc ambele cu 100%, iar rezistența la magie crește și ea cu 50%. Runa nu poate fi utilizată asupra dragonilor.

fi construite și de dwarfi), există cinci niveluri pentru altarele runice.

Eroii pot folosi în luptă runele; pentru a le utiliza, ei nu consumă mana, ci resurse.

De fiecare dată când este rândul unei unități, poate fi utilizată o rună. În principiu, o anumită rună poate fi utilizată o singură dată asupra unei unități în timpul unei lupte. Există o abilitate specială a eroilor care permite utilizarea unei rune de două ori, dar la a doua utilizare prețul crește. În casetă sunt descrise runele disponibile.

Eroii

Evident, specialitatea rasială a eroilor dwarfi constă în utilizarea runelor.

Ei se concentrează în principal asupra tehnicilor defensive și asupra utilizării forței magice.

În timpul campaniei apar patru eroi dwarfi.

Primul dintre ei este *Wulfstan*, personajul principal al campaniei, membru al clanului *Winterwind*.

Al doilea erou este *Rolf*, fratele vitreg al lui *Wulfstan*; planul său secret este acela de a deveni rege.

Cel de-al treilea erou este actualul rege *Tolghar*. Îi lipsește încrederea în sine și consilierii săi îl manipulează, profitând de temerile regelui. Va fi adversarul lui *Wulfstan* în ultima bătălie a campaniei cu dwarfii.

Cel de-al patrulea erou care apare în campanie este *Hangvul*. El consideră că este singurul căruia i se datorează păstrarea integrității regatului și consideră că sarcina sa (dată de dragoni) este aceea de a se asigura că regele *Tolghar* îi asculta sfaturile. *Hangvul* și-a pierdut un ochi și o mână în războaiele cu elfii întunecați și și-a creat singur altele noi, folosind oțel și magie, ceea ce a avut efecte importante asupra personalității sale.

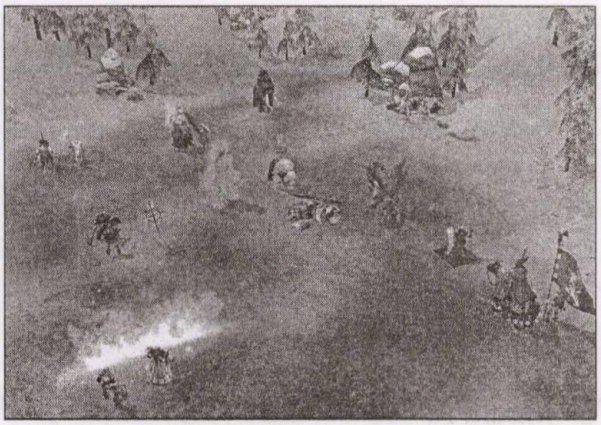
Renegații

Hammers of Fate introduce șapte upgrade-uri alternative pentru unitățile *haven*. Noile unități nu pot fi recrutate în castele, dar au un rol important în poveste; prima campanie a *add-on*-ului le este dedicată.

Sunt creaturi malefice, caracteristicile lor demonstrând acest lucru. Ele reprezintă forțele pe care se bazează noua conducere a *Imperiului Griffinului*.

Caracteristicile celor șapte noi unități (*red haven*) sunt următoarele:

- *Enforcer:*
  - *Atac*: 2;
  - *Apărare*: 1;
  - *Interval pentru daunele produse*: 1 - 2;
  - *Inițiativă*: 8;
  - *Viteză*: 4;
  - *Vitalitate*: 6;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 30 aur;
  - *Rata de creștere săptămânală*: 22;
  - *Experiență obținută de adversar*: 8;
  - *Evaluarea puterii*: 72;
  - *Abilități speciale:*
    - ◊ *Taxpayer:* în fiecare zi jucătorul primește o cantitate de aur egală cu numărul total al acestor unități, aflate sub controlul său;
    - ◊ *Bash:* fiecare atac are șansa de a împietri inamicul; acesta nu ripostează și își pierde inițiativa;
- *Crossbowman:*
  - *Atac*: 5;
  - *Apărare*: 4;
  - *Interval pentru daunele produse*: 2 - 8;
  - *Inițiativă*: 8;
  - *Viteză*: 4;
  - *Vitalitate*: 10;
  - *Mana*: 0;
  - *Lovituri la distanță*: 10;
  - *Cost*: 80 aur;
  - *Rata de creștere săptămânală*: 12;
  - *Experiență obținută de adversar*: 20;
  - *Evaluarea puterii*: 203;



  - *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *No Range Penalty:* unitatea nu suferă penalizări în cazul atacurilor la o distanță mai mare decât jumătate din dimensiunea câmpului de luptă;
- *Vindicator:*
  - *Atac*: 8;
  - *Apărare*: 8;
  - *Interval pentru daunele produse*: 2 - 5;
  - *Inițiativă*: 8;
  - *Viteză*: 4;
  - *Vitalitate*: 26;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 130 aur;
  - *Rata de creștere săptămânală*: 10;
  - *Experiență obținută de adversar*: 29;
  - *Evaluarea puterii*: 299;
  - *Abilități speciale:*
    - ◊ *Cleave:* în cazul în care atacul duce cel puțin la uciderea unui adversar, unitatea beneficiază de un nou atac împotriva aceluiași inamic;
- *Battle Griffin:*
  - *Atac*: 6;
  - *Apărare*: 12;
  - *Interval pentru daunele produse*: 5 - 15;
  - *Inițiativă*: 15;
  - *Viteză*: 7;
  - *Vitalitate*: 35;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 360 aur;
  - *Rata de creștere săptămânală*: 5;
  - *Experiență obținută de adversar*: 59;
  - *Evaluarea puterii*: 725;

- *Abilități speciale:*
  - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
  - ◊ *Flyer:* unitatea se poate deplasa ignorând obstacolele din calea sa;
  - ◊ *Unlimited Retaliation*: unitatea ripostează la toate atacurile, cu excepția celor efectuate de unități care au abilitatea *No Enemy Retaliation*;
  - ◊ *Battle Frenzy*: fiecare ripostă a unității este de 1.5 ori mai puternică decât precedenta;

- *Zealot:*
  - *Atac*: 20;
  - *Apărare*: 14;
  - *Interval pentru daunele produse*: 9 - 12;
  - *Inițiativă*: 10;
  - *Viteză*: 5;
  - *Vitalitate*: 80;
  - *Mana*: 15;
  - *Lovituri la distanță*: 5;
  - *Cost*: 900 aur;
  - *Rata de creștere săptămânală*: 3;
  - *Experiență obținută de adversar*: 100;
  - *Evaluarea puterii*: 1523;
  - *Abilități speciale:*
    - ◊ *Shooter:* poate provoca daune de la distanță; dacă distanța până la țintă este mai mare decât jumătate din dimensiunea câmpului de luptă, atunci ținta suferă doar jumătate din daune; unitatea nu își poate folosi atacul la distanță dacă este blocată de o unitate adversă; dacă nu există un cărucior cu muniție, numărul loviturilor la distanță este limitat de muniția disponibilă;
    - ◊ *No Melee Penalty*: unitatea nu este penalizată în timpul atacului corp la corp;
    - ◊ *Caster:* unitatea poate folosi anumite vrăji, în mod asemănător eroilor; cantitatea de mana este limitată, dar este întotdeauna la nivelul maxim la începutul luptei; puterea magică depinde de numărul creaturilor;
    - ◊ *Cartea cu vrăji:*
      - → *Blindness* (10 mana): orbește inamicul selectat; adversarul nu se poate mișca, nu poate ataca și nu își poate folosi abilitățile; efectul se pierde dacă unitatea afectată este atacată; vraja nu poate fi utilizată împotriva unităților *undead*, a celor elementale și a celor mecanice;
      - → *Righteous Might* (6 mana): unitatea aleasă va deveni plină de furie și va fi însetată de sânge, ceea ce duce la creșterea atacului;

- *Champion:*
  - *Atac*: 24;
  - *Apărare*: 20;
  - *Interval pentru daunele produse*: 20 - 35;
  - *Inițiativă*: 12;
  - *Viteză*: 8;
  - *Vitalitate*: 100;
  - *Mana*: 0;

Renegații

Enforcer
Uneori este nevoie de puțină brutalitate pentru a realiza ceea ce trebui, cel puțin așa cred enforcer-ii. Sunt trupele de bază ale noului regim din imperiu și nu ezită să își năucească adversarii folosind lovituri cu scutul.

Crossbowman
Noua guvernare a adus tehnologii noi și echipamente îmbunătățite, ceea ce a permis imperiului să folosească soldați dotați cu arbalete. Noii arcași pot să își repete rapid atacurile.

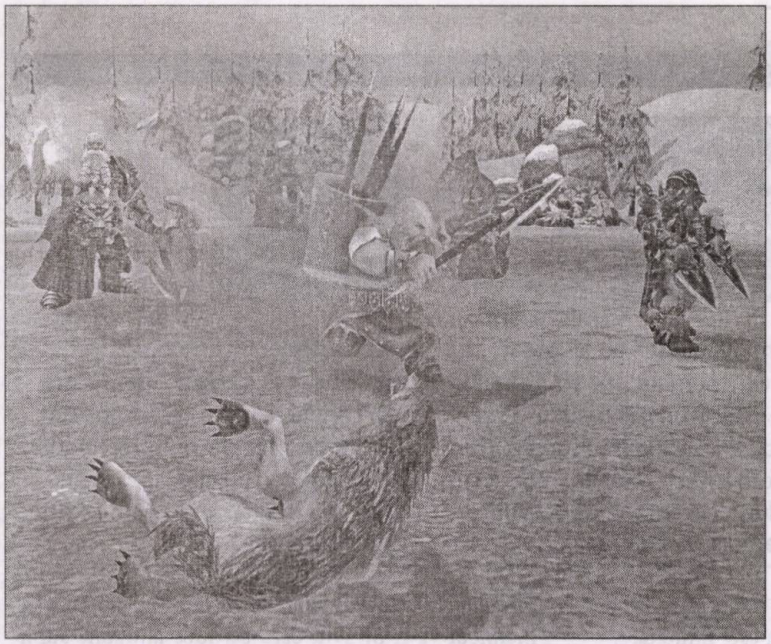
Vindicator
Vindicatorii sunt războinici conștienți de avantajele pe care le aduce zelul religios. Devotamentul i-a făcut să învețe să își îmbunătățească atacurile împotriva celor care îi înfruntă.

Battle Griffin
Ajunse în pragul nebuniei datorită regimului brutal de antrenamente la care sunt supuse de către stăpâni, aceste creaturi intră în luptă cu o frenezie devastatoare.

Zealot
Fanatismul i-a făcut pe acești foști preoți ai luminii să treacă de partea noii conduceri. La fel ca preoții din rândurile cărora provin, își folosesc atacurile magice pentru a-și devasta inamicii și aura sfântă pentru a-și încuraja aliații.

Champion
Fanatici care au depus un jurământ de loialitate în fața Sfintei Isabel, sunt unități uriașe, care folosesc armuri grele. Tactica lor favorită constă în a-și lăsa toată greutate asupra adversarilor.

Fallen Angel
Capriciile zeilor sunt de neînțeles. Putem doar ghici de ce acești îngeri au decăzut din grația divină. În urma crudelor lecții pe care le-au învățat datorită decăderii, ei au reușit să dobândească puterea de a prelua vitalitatea adversarilor.



- *Lovituri la distanță*: 0;
- *Cost*: 1550 aur;
- *Rata de creștere săptămânală*: 2;
- *Experiență obținută de adversar*: 160;
- *Evaluarea puterii*: 2520;
- *Abilități speciale:*
  - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
  - ◊ *Champion Charge:* dacă unitatea parcurge mai mult de două celule pentru a ataca, atunci este afectată și unitatea aflată imediat în spatele celei atacate (daunele suferite de a doua unitate reprezintă jumătate din daunele suferite de ținta principală a atacului);

- *Fallen Angel:*
  - *Atac*: 35;
  - *Apărare*: 25;
  - *Interval pentru daunele produse*: 25 - 75;
  - *Inițiativă*: 11;
  - *Viteză*: 8;
  - *Vitalitate*: 220;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 4200 aur;
  - *Rata de creștere săptămânală*: 1;
  - *Experiență obținută de adversar*: 322;
  - *Evaluarea puterii*: 6003;
  - *Abilități speciale:*
    - ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
    - ◊ *Flyer:* unitatea se poate deplasa ignorând obstacolele din calea sa;
    - ◊ *Life Drain:* unitatea poate prelua vitalitatea unității atacate, readucând la viață o parte din trupele pierdute; vitalitatea recuperată este egală cu jumătate din daunele produse.

Unități neutre

Add-on-ul ne oferă și trei unități neutre, al căror rol în poveste este destul de redus. Caracteristicile acestora sunt următoarele:

- *Wolf:*
  - *Atac*: 5;
  - *Apărare*: 3;
  - *Interval pentru daunele produse*: 3 - 5;
  - *Inițiativă*: 14;
  - *Viteză*: 6;
  - *Vitalitate*: 25;
  - *Mana*: 0;
  - *Lovituri la distanță*: 0;
  - *Cost*: 150 aur;
  - *Rata de creștere săptămânală*: 8;
  - *Experiență obținută de adversar*: 31;
  - *Evaluarea puterii*: 355;
  - *Abilități speciale:*
    - ◊ *Pack Hunter:* când unitatea atacă un adversar, acesta este atacat de toate celelalte unități de același tip care pot efectua un atac;
    - ◊ *Howl:* în timpul luptei, o singură dată, este invocat un grup de unități aliate cu o putere egală;

- *Mummy:*
  - *Atac*: 8;
  - *Apărare*: 9;
  - *Interval pentru daunele produse*: 20 - 30;
  - *Inițiativă*: 15;
  - *Viteză*: 3;
  - *Vitalitate*: 50;
  - *Mana*: 32;
  - *Lovituri la distanță*: 0;
  - *Cost*: 900 aur;
  - *Rata de creștere săptămânală*: 3;
  - *Experiență obținută de adversar*: 112;
  - *Evaluarea puterii*: 1542;
  - *Abilități speciale:*
    - ◊ *Undead:* la fel ca toate creaturile din necropole, mumiile nu sunt afectate de moral; de asemenea, nu pot fi afectate de otrăvire, orbire sau de alte vrăji care afectează mintea;
    - ◊ *Hexing Attack:* după un atac, inamicul va suferi unul dintre următoarele efecte: *weakness, slow, suffering* sau *vulnerability* (puterea acestora este aceiași cu cea a vrăjilor corespunzătoare, la nivel de expert);

Unități neutre

Wolf
Lupii sunt bestii periculoase și însetate de sânge care atacă în haită și își folosesc cu inteligență abilitățile. Dacă un grup atacă, atunci toate celelalte vor ataca aceeași țintă, dacă o vor putea ajunge. Mai mult, înainte de a ataca, lupii produc un urlet puternic prin care își cheamă ajutoare.

Mummy
Lești readuse la viață ale vechilor vrăjitori, mumiile reprezintă un chin pentru orice armată. După ce își atacă inamicii, mumiile pot să îi afecteze pe aceștia cu diverse vrăji din școala magiei întunecate. De asemenea, mumiilor pot să readucă la viață o parte dintre creaturile aliate ucise.

Manticore
Creaturi intimidante care au corp de leu, aripi de dragon și coadă de scorpion, sunt oponenți redutabili. În luptă își pot depăși repede adversarii și îi pot otrăvi folosind un venin atât de puternic încât substanțele folosite de asasini par, prin compatație, simple medicamente.

◊ *Caster:* unitatea poate folosi anumite vrăji, în mod asemănător eroilor; cantitatea de mana este limitată, dar este întotdeauna la nivelul maxim la începutul luptei; puterea magică depinde de numărul creaturilor;
◊ *Cartea cu vrăji:*
→ *Confusion* (9 mana): o parte dintre unitățile afectate vor uita ce trebuie să facă, nu își vor putea folosi atacul la distanță și nu vor risposta când sunt atacate; vraja nu poate fi utilizată împotriva unităților *undead*, a celor elementale și a celor mecanice;
→ *Raise Dead* (9 mana): reanimează creaturile dintr-un grup aliat; efectul este permanent dacă este aplicată unui grup de creaturi *undead* și temporar (doar până la sfârșitul luptei) pentru celelalte creaturi; fiecare utilizare a vrăjii reduce cu 20% vitalitatea grupului afectat;

- *Manticore:*
  - ♦ *Atac*: 15;
  - ♦ *Apărare*: 15;
  - ♦ *Interval pentru daunele produse*: 30 - 50;
  - ♦ *Inițiativă*: 9;
  - ♦ *Viteză*: 5;
  - ♦ *Vitalitate*: 120;
  - ♦ *Mana*: 0;
  - ♦ *Lovituri la distanță*: 0;
  - ♦ *Cost*: 1800 aur;
  - ♦ *Rata de creștere săptămânală*: 2;
  - ♦ *Experiență obținută de adversar*: 164;
  - ♦ *Evaluarea puterii*: 2523;
  - ♦ *Abilități speciale:*
    ◊ *Large Creature:* unitatea ocupă patru celule (dispuse 2 × 2) pe câmpul de luptă;
    ◊ *Flyer:* unitatea se poate deplasa ignorând obstacolele din calea sa;
    ◊ *Poisonous Attack:* pe lângă daunele provocate de atac, inamicul va fi otrăvit, ceea ce duce la daune suplimentare timp de trei ture; valoarea daunelor depinde de numărul creaturilor care atacă.

La final

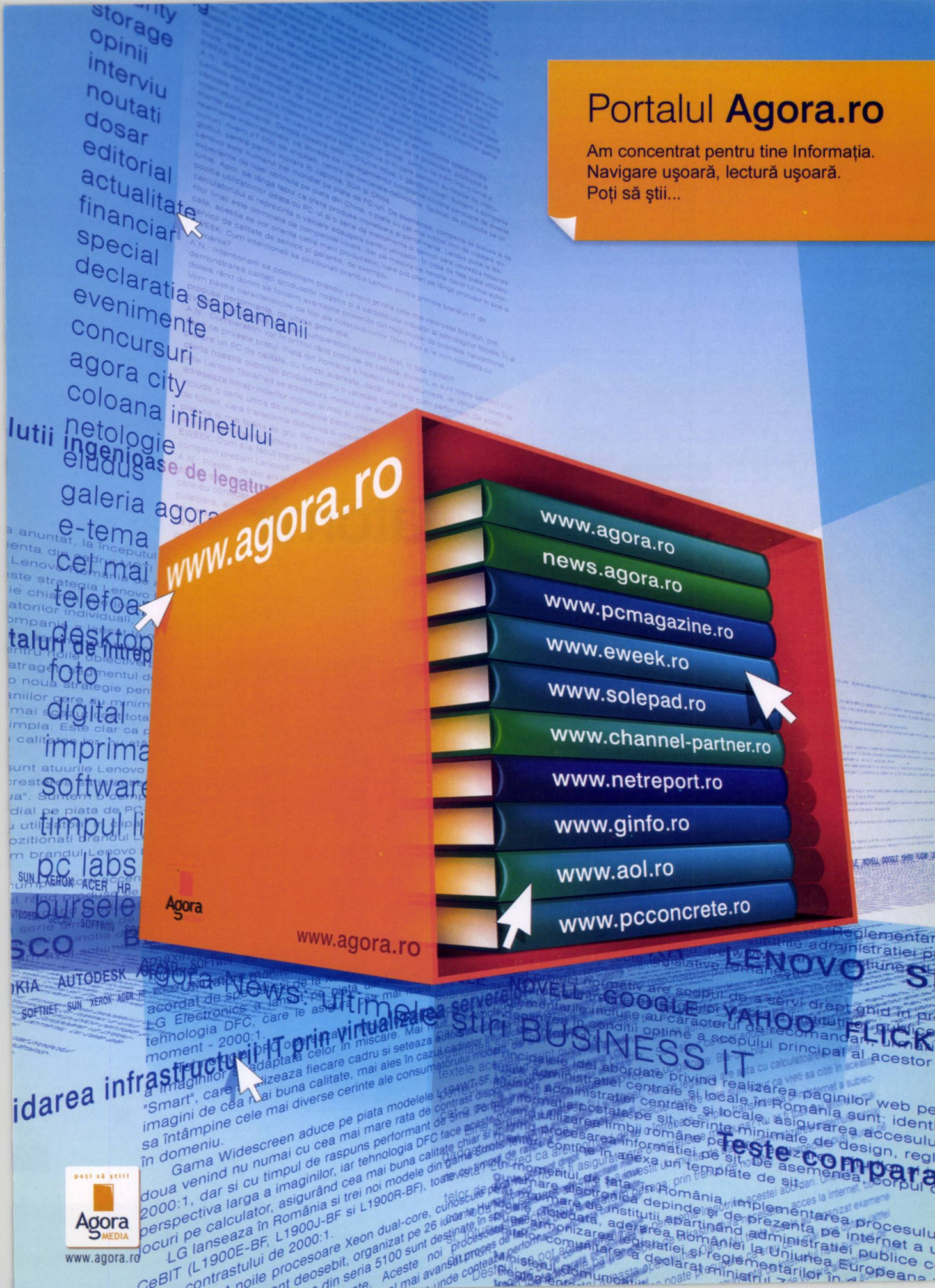
Mai trebuie menționate celelalte noutăți importante introduse odată cu acest *add-on*:

- generator aleator de hărți;
- runde simultane în *multiplayer*, până în momentul în care ariile de interes ale jucătorilor se intersectează;
- noi hărți;
- noi artefacte;
- caravane care permit transportul rapid de trupe între orașe, precum și transportul trupelor de la clădirle externe înspre orașe.

Povestea continuă din momentul în care s-a oprit în jocul original. Vom începe cu *Freyda*, fiica lui *Godric*, care își va da seama că ceva nu este în regulă cu noua conducere a imperiului. Îl vom ajuta apoi pe *Wulfstan* să aducă darfii pe calea cea bună. În final, în rolul elfei întunecate *Ylaya* ne vom întâlni cu toate personajele pozitive pentru a face față forțelor răului. Pe parcurs, vom afla că *Isabel* nu este deloc o creatură malefică, așa cum s-ar fi crezut la început. Deși demonii nu au un rol important în poveste, ne vom întâlni și cu o succubus care va continua îndeplinirea sarcinilor date de suveranul demonilor.

Mai trebuie menționat faptul că, între timp, a apărut un patch care aduce modificări esențiale cu privire la modul în care funcționează necromanția, precum și antrenamentele oamenilor.

De asemenea, există și o versiune *Silver Edition*, care conține jocul original, împreună cu *add-on*-ul.


Portalul Agora.ro
Am concentrat pentru tine Informația.
Navigare ușoară, lectură ușoară.
Poți să știi...
www.agora.ro
www.agora.ro
news.agora.ro
www.pcmagazine.ro
www.eweek.ro
www.solepad.ro
www.channel-partner.ro
www.netreport.ro
www.ginfo.ro
www.aol.ro
www.pcconcrete.ro
Agora
www.agora.ro
Agora MEDIA
www.agora.ro

Cabinetul
Călin Tatomir

Noi ştim cum

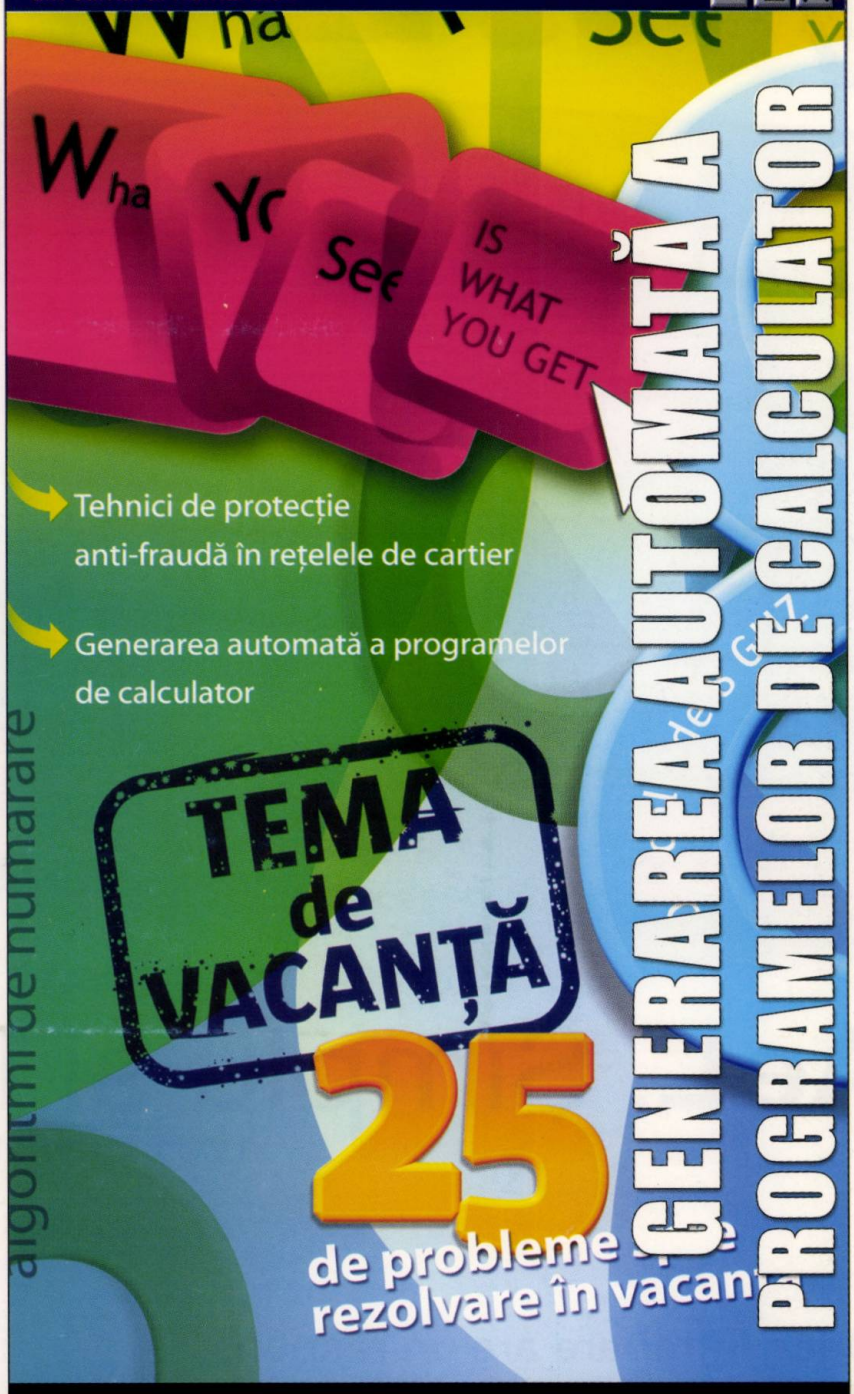
gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

Mai 2007
Vol. 17/5

40.000 LEI / 4 LEI NOI

din sumarul numărului:



actual
Dincolo de 8 GHZ

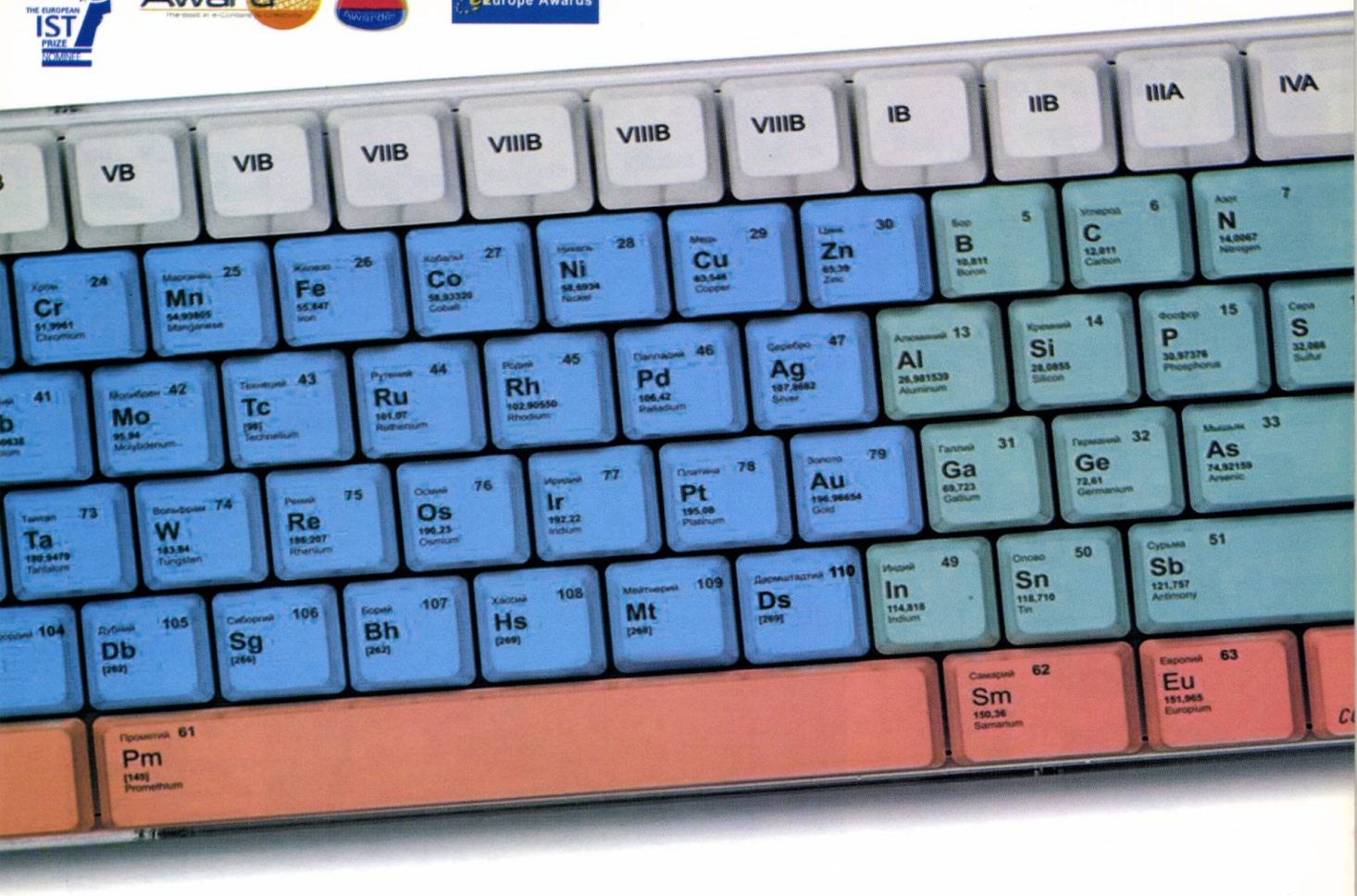
focus
Variante recursive

serial
Rețele de cartier

internet
Editor WYSIWYG într-o pagină WEB

5 948373 000645 05

O publicație Agora Media



AeL. Simplu şi la obiect.

Lecţii interactive pentru noua generaţie.

Orice întrebare are un răspuns!

Noutăţi, suport tehnic, competiţii,
forum : www.portal.edu.ro

Orice problemă are o soluţie!

Centrul de suport tehnic de luni până
vineri între orele 8.00-20.00
Telverde : 0800410444 (linie gratuită)

Orice cauţi poate fi găsit!

Ai nevoie de o anumită lecţie electronică?
Întâi de toate, caut-o în biblioteca virtuală
AeL. Odată găsită, descarc-o gratuit!

AeL
Built to teach intelli-gently

sei

SIVECO
ROMANIA SA

Vol. 17, Nr. 5 /2007

gazeta de informatică

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tiberiu Socaciu
Colaboratori:
Vlad Bazon
Oana Muntean
Ion Petre

Adresa pentru corespondență:
Computer Libris Agora,
Șerpuitoare 25, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/534.911, 0265/266.516
Fax: 0265/266.290

http://www.ginfo.ro
ISSN: 1222-7129
Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL

Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NET Report*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

O publicație Agora MEDIA

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu

Difuzare:
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

sumar

actual.. PAG. 5

Dincolo de 8 GHZ

Mihai SCORȚARU

probleme.. PAG. 6

TEMA DE VACANȚĂ

Mihai SCORȚARU

soluții.. PAG. 21

Asupra unei PROBLEME de la ONI

Ion PETRE

e-learning.. PAG. 23

Intel® Teach PAG. 23

R. JUGUREANU, F. ILIA

NetOp® School PAG. 39

focus.. PAG. 24

Demonstrarea CORECTITUDINII variantei RECURSIVE

Tiberiu SOCACIU

Algoritmi de NUMĂRARE

Ion PETRE

internet.. PAG. 33

ÎNCORPORAREA unui editor WYSIWYG într-o pagină WEB

Vlad BAZON

serial.. PAG. 40

Tehnici de PROTECȚIE ANTI-FRAUDĂ în rețelele de cartier

Tiberiu SOCACIU

babel.. PAG. 42

Generarea AUTOMATĂ a PROGRAMELOR de calculator

Oana MUNTEAN

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

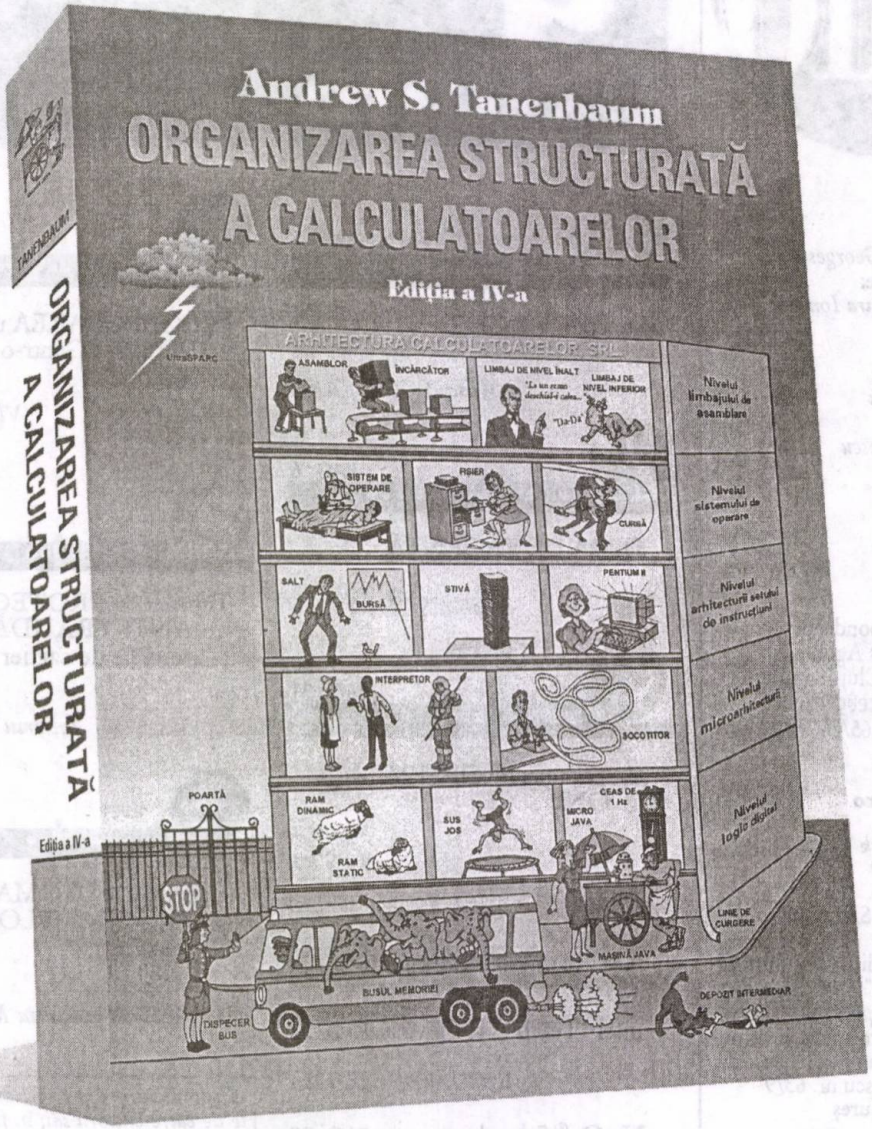
Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

Singura condiție pe care trebuie să o îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre.

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii. Pentru mai multe detalii vizitați:
http://www.ginfo.ro/revista/publicare.shtml

Cine are carte, are parte!

...de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețelele de calculatoare

- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o REDUCERE semnificativă!

Byblos srl, tel: 0318-056-314, 0721-927-499, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librărie: Galeria NOI-La Calu' Bălan, Str. Halelor Nr. 3 (după Hanul lui Manuc).

PAYPAL în România

Începând din mai 2007, serviciul de plăți online PayPal permite utilizatorilor din România să beneficieze de facilitățile oferite. Deși deocamdată conturile românilor pot fi folosite doar pentru plăți, cu siguranță numărul cumpărăturilor online efectuate de români va crește în scurt timp.

De multe ori ne-am fi dorit să cumpărăm diverse lucruri de pe internet și ne-am lovit de tot felul de bariere. Observam că nu avem carte de credit, mergeam la bancă, făceam o cerere, deschideam un cont, plăteam comisioane, primeam după câteva săptămâni cardul și încercam din nou.

Cu cardul îm mână fugeam la calculator, alegeam ceea ce doream, completam datele cardului și eram informați că tranzacția a eșuat. Încercam încă odată și se întâmpla același lucru. Dădeam telefon la bancă și aflam că, de fapt, cardul nostru nu poate fi folosit decât în *România*.

Făceam o nouă cerere, așteptam din nou, plăteam alte comisioane și ajungeam din nou în fața ecranului și priveam textul care ne spunea că ceva nu e în regulă. Sunam din nou la bancă și aflam că, deși cu magnificul card puteam să folosim bancomatele din *Tunisia*, *Kazahstan* sau *Noua Zeelandă*, cardul nostru nu poate fi folosit pentru tranzacții online.

După o altă cerere, alte săptămâni de așteptare și alte comisioane plătite, aveam cardul "cel bun". Totul funcționa perfect până în momentul în care site-ul de pe care făceam cumpărăturile ne informa că nu acceptă carduri din România.

În acest moment, de obicei, renunțam și așteptam ca produsul să apară și la noi.

Totuși, am fi putut observa printre metodele de plată așa-numitul *PayPal*.

Dacă am fi studiat "problema" am fi aflat că acesta este un serviciu online care, practic, oferă un cont virtual oricărei persoane care deține o adresă de e-mail.

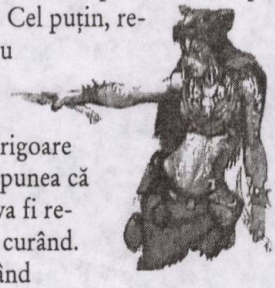
Ar fi fost frumos să putem folosi așa-ceva. Unii găseau soluții ingenioase... Fratele, cumnatul sau prietenul de peste ocean putea folosi *PayPal* și, dacă îl rugam frumos, ne ajuta...

Dar, în anul de grație 2007, *PayPal* s-a gândit că România nu mai face parte din lumea a șaptea și a permis accesul utilizatorilor români. Totuși, să nu credeți că acest fapt se datorează integrării în *Uniunea Europeană*, aderării la *NATO* sau alte chestii de acest gen (nici măcar suspendările și referendumurile nu au nimic de-a face cu disponibilitatea *PayPal* în România).

România, a fost inclusă într-un grup de câteva zeci de țări și astfel a ajuns pe lista țărilor din care se pot face plăți. Deocamdată, nu putem și primi bani folosind doar o adresă de e-mail.

Faptul că *Uniunea Europeană* nu prea are nimic de-a face cu acest "eveniment" este "demonstrat" de faptul că, în dreptul țării noastre, pe site-ul *PayPal* apărea steagul *Republicii Socialiste România*. Dacă nu știați sau ați uitat cum arăta stema *R.S.R.*, cei de la *PayPal* v-ar fi adus aminte.

Probabil mulți români au trimis mesaje revoltate, dar drapelul nu a dispărut decât după câteva zile. Cel puțin, reclamanții au primit mesaje cu scuzele de rigoare în care se spunea că problema va fi remediată în curând.



Până când numărul tranzacțiilor prin *PayPal* ale românilor va rivaliza cu cel al tranzacțiilor din țările din jur, vă spun că orcii reprezintă nivelurile de dificultate ale articolelor din acest număr și, vă urez o vacanță plăcută și îmi exprim, ca la fiecare sfârșit de sezon, speranța că ne vom "revedea" la toamnă.

Mihai Scorțaru

Cea mai bună introducere din lume în rețelele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine

EDIȚIA A PATRA

Rețele de calculatoare

ANDREW S. TANENBAUM

Cea mai bună introducere în arhitectura calculatoarelor - complet actualizată, reflectă ultimele schimbări din dinamica lume a calculatoarelor

Andrew S. Tanenbaum

ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR

Ediția a IV-a

SISTEME DE OPERARE MODERNE

EDIȚIA A DOUA

Andrew S. Tanenbaum

Această carte de referință, bazată pe exemple concrete, oferă informații practice de primă mână pentru înțelegerea și construirea sistemelor de operare moderne.

THOMAS H. CORMEN CHARLES E. LEISERSON RONALD R. RIVEST

INTRODUCERE ÎN ALGORITMI

Cartea tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, fiind binevenita elevilor, studenților și profesorilor

CĂRȚILE ESENȚIALE ▸

...cărțile care îți însoțesc cariera

Comandă-le pe toate și le iei cu 20% mai ieftin.
Iar dacă ești student, cu 30%.

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

Dincolo de 8 GHZ

Mihai Scorțaru

Acum câțiva ani exista un adevărat război între Intel și AMD în ceea ce privește frecvențele procesoarelor. După ce AMD atinsese prima viteza de 1 GHz în anul 2000, Intel a răspuns atingând prima vitezele de 2 și 3 GHz. Dar, cursa s-a oprit datorită diverselor limitări tehnologice. Intel nu a mai atins 4 GHz (cel mai rapid procesor Intel funcționează la 3.8 GHz), iar AMD a atins de-abia recent bariera de 3 GHz. Între timp, pentru a mări performanțele procesoarelor, cele două companii s-au concentrat asupra altor aspecte cum ar fi dimensiunile memoriei cache, integrarea mai multor core-uri etc. Totuși, entuziaștii își doresc viteze din ce în ce mai mari și, folosind metode de răcire extravagante, ating frecvențe impresionante. Recent, italienii de la OC Team au reușit să convingă un procesor Intel Pentium 4 să ruleze la viteza de 8.18 GHz.

Italienii de la *OC Team* au reușit să atingă bariera de 8 GHz la începutul acestui an.

Atunci, ei folosiseră un procesor *Intel Pentium* 4, modelul 631, procesor care rulează în mod normal la o frecvență de 3 GHz.

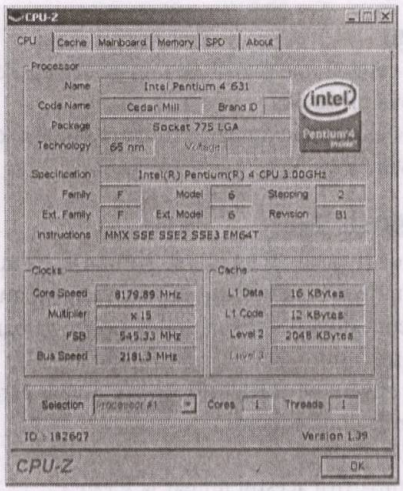
Folosind pentru răcire nitrogen lichid și jucându-se puțin cu voltajele, ei reușiseră atunci să modifice frecvența de bază a magistralei de la 200 MHz la 533.34 MHz. Păstrând multiplicatorul la valoarea 15 (multiplicatorul este blocat), aceasta a dus la o viteză de 8001 MHz.

Succesul lor este ilustrat de imaginea următoare:

După atingerea acestui prag psihologic (practic, s-a reușit dublarea vitezei față de orice procesor *Intel* disponibil pe piață), italienii au continuat să experimenteze și au reușit să mărească mai mult viteza magistralei, până la valoarea de 545.33 MHz. Aceasta a făcut ca viteza procesorului să crească și ea cu 179 MHz, atingând valoarea de 8180 MHz.

Aceasta înseamnă o supratactare de 173%, performanță care mai fusese obținută doar pentru procesoare din familia Celeron, care rulau, în mod normal, la frecvențe de 1.4 - 1.5 GHz.

Acest ultim succes este prezentat în următoarea imagine:



Încercați-vă puterile!

Tema de VACANȚĂ

Mihai Scorțaru

În cele ce urmează vă propunem spre rezolvare 25 de probleme, sperăm noi interesante, care ar putea fi folosite în pregătirea pentru concursurile din următorul an școlar.

Clasa a V-a

P070501: Un multiplu

Se citesc de la tastatură un număr natural *n*, cel mult egal cu 10000, precum și un număr strict pozitiv *k*, cel mult egal cu 100. Să se afișeze pe ecran cel mai mic număr care satisface următoarele două condiții:
- este cel puțin egal cu *n*;
- este divizibil cu *k*.

Exemple

Intrare
```
24
57
```
Ieșire
```
57
```
Intrare
```
13
3
```
Ieșire
```
15
```

Explicații
- 57 este cel mai mic număr, mai mare sau egal cu 24, care este divizibil cu 57;
- 15 este cel mai mic număr, mai mare sau egal cu 13, care este divizibil cu 3.

P070502: Rest

Se citesc de la tastatură un număr natural *n*, cel mult egal cu 10000, un număr natural strict pozitiv *k*, cel mult egal cu 100, precum și un număr natural *r*, a cărui valoare este cuprinsă între 0 și *k* - 1.

Să se afișeze pe ecran cel mai mic număr care satisface următoarele două condiții:
- este cel puțin egal cu *n*;
- restul împărțirii acestui număr la *k* este *r*.

Exemple

Intrare
```
24
57
0
```
Ieșire
```
57
```
Intrare
```
13
3
2
```
Ieșire
```
14
```

Explicații
- 57 este cel mai mic număr, mai mare sau egal cu 24, pentru care restul împărțirii la 57 este 0;
- 14 este cel mai mic număr, mai mare sau egal cu 13, pentru care restul împărțirii la 3 este 2.

P070503: Suma cifrelor

Se citește de la tastatură un număr natural *n*, cel mult egal cu 10000.

Să se afișeze pe ecran valoarea absolută a diferenței dintre suma cifrelor de pe pozițiile impare și suma cifrelor de pe pozițiile pare. Considerăm că poziția ultimei cifre este 1, poziția penultimei cifre este 2 și așa mai departe.

Exemple

Intrare
```
2457
```
Ieșire
```
4
```
Intrare
```
777
```

Ieșire
7

Intrare
1234

Ieșire
2

Explicații
- pentru numărul 2457 suma cifrelor de pe pozițiile impare este 4 + 7 = 11, iar suma cifrelor de pe pozițiile pare este 2 + 5 = 7; valoarea absolută a diferenței dintre cele două sume este |11 - 7| = 4;
- pentru numărul 777 suma cifrelor de pe pozițiile impare este 7 + 7 = 14, iar suma cifrelor de pe pozițiile pare este 7 (avem o singură cifră 7 pe pozițiile pare); valoarea absolută a diferenței dintre cele două sume este |14 - 7| = 7;
- pentru numărul 1234 suma cifrelor de pe pozițiile impare este 1 + 3 = 4, iar suma cifrelor de pe pozițiile pare este 2 + 4 = 6; valoarea absolută a diferenței dintre cele două sume este |4 - 6| = 2.

P070504: Număr extrem

Spunem că un număr este *extrem* dacă suma cifrelor sale este mai mare decât produsul cifrelor sale.

Se citește de la tastatură un număr natural *n*, cel mult egal cu 10000.

Să se afișeze pe ecran mesajul DA în cazul în care numărul *n* este extrem și mesajul NU în caz contrar.

Exemple
Intrare
2457

Ieșire
NU

Intrare
13

Ieșire
DA

Intrare
123

Ieșire
NU

Explicații
- suma cifrelor numărului 2457 este 2 + 4 + 5 + 7 = 18, iar produsul cifrelor sale este 2 · 4 · 5 · 7 = 280; datorită faptului că 18 este mai mic decât 280, numărul 2457 nu este extrem;
- suma cifrelor numărului 13 este 1 + 3 = 4, iar produsul cifrelor sale este 1 · 3 = 3; datorită faptului că 4 este mai mare decât 3, numărul 13 este extrem;
- suma cifrelor numărului 123 este 1 + 2 + 3 = 6, iar produsul cifrelor sale este 1 · 2 · 3 = 6; datorită faptului că cele două valori sunt egale (deci suma cifrelor nu este mai mare decât produsul lor), numărul 123 nu este extrem.

Clasa a VI-a

P070505: Text echilibrat

Spunem că un text este echilibrat dacă numărul vocalelor este cel mult egal cu dublul numărului consoanelor, iar numărul consoanelor este cel mult egal cu dublul numărului vocalelor.

De la tastatură se va citi un text care va conține cel mult 100 de caractere, printre care se vor afla cel puțin o vocală și cel puțin o consoană.

Să se afișeze pe ecran mesajul DA în cazul în care textul este echilibrat și mesajul NU în caz contrar.

Exemple
Intrare
Gazeta de Informatica

Ieșire
DA

Intrare
Monstru

Ieșire
NU

Explicații
- textul Gazeta de Informatica conține 9 vocale și 10 consoane; datorită faptului că 9 < 2 · 10 = 20 și 10 < 2 · 9 = 18, textul este echilibrat.
- textul Monstru conține 2 vocale și 5 consoane; datorită faptului că 2 < 2 · 5 = 10, dar 5 > 2 · 2 = 4, textul nu este echilibrat.

P070506: Cifre, litere, ...

De la tastatură se va citi un șir format din cel mult 200 de caractere. Să se determine și să se afișeze pe ecran câte dintre caracterele șirului sunt cifre, câte sunt litere ale alfabetului englezesc și câte sunt alte caractere.

Exemple
Intrare
GInfo * 2457

Ieșire
5
4
3

Intrare
```
Gazeta de Informatica
```

Ieşire
```
19
0
2
```

P070507: Şir de numere

Se citeşte de la tastatură un şir format din n ($5 < n < 1000$) numere întregi pozitive, cel mult egale cu 30000.

Să se determine şi să se afişeze pe ecran numărul de numere care conţin o cifră, numărul de numere care conţin două cifre, numărul de numere care conţin trei cifre, numărul de numere care conţin patru cifre şi numărul de numere care conţin cinci cifre.

Exemple

Intrare
```
10
87 123 2457 3 0 10 100 10000 22222 13
```

Ieşire
```
2
3
2
1
2
```

Intrare
```
10
1 2 3 4 5 6 7 8 9 10
```

Ieşire
```
9
1
0
0
0
```

Explicaţii

Pentru primul exemplu:
- cele două numere formate dintr-o singură cifră sunt 3 şi 0;
- cele trei numere formate din două cifre sunt 87, 10 şi 13;
- cele două numere formate din trei cifre sunt 123 şi 100;
- numărul format din patru cifre este 2457;
- cele două numere formate din cinci cifre sunt 10000 şi 22222.

Pentru al doilea exemplu:
- primele nouă numere sunt formate din câte o cifră;
- al zecelea număr este format din două cifre;
- nu există numere formate din trei cifre;
- nu există numere formate din patru cifre;
- nu există numere formate din cinci cifre.

P070508: Şir perfect

Se citeşte de la tastatură un şir format din n ($5 < n < 500$) numere întregi pozitive, cel mult egale cu 10000.

Spunem că şirul este *perfect* dacă toate elementele şirului au aceeaşi sumă a cifrelor.

Să se afişeze pe ecran mesajul DA în cazul în care şirul este perfect şi mesajul NU în caz contrar.

Exemple

Intrare
```
5
2457 99 5445 1818 9009
```

Ieşire
```
DA
```

Intrare
```
11
9 18 27 36 45 54 63 72 81 90 99
```

Ieşire
```
NU
```

Explicaţii

Pentru primul exemplu:
- suma cifrelor numărului 2457 este 2 + 4 + 5 + 7 = 18;
- suma cifrelor numărului 99 este 9 + 9 = 18;
- suma cifrelor numărului 5445 este 5 + 4 + 4 + 5 = 18;
- suma cifrelor numărului 1818 este 1 + 8 + 1 + 8 = 18;
- suma cifrelor numărului 9009 este 9 + 0 + 0 + 9 = 18.

Pentru al doilea exemplu:
- suma cifrelor numărului 9 este 9;
- suma cifrelor numărului 18 este 1 + 8 = 9;
- suma cifrelor numărului 27 este 2 + 7 = 9;
- suma cifrelor numărului 36 este 3 + 6 = 9;
- suma cifrelor numărului 45 este 4 + 5 = 9;
- suma cifrelor numărului 54 este 5 + 4 = 9;
- suma cifrelor numărului 63 este 6 + 3 = 9;
- suma cifrelor numărului 72 este 7 + 2 = 9;
- suma cifrelor numărului 81 este 8 + 1 = 9;
- suma cifrelor numărului 90 este 9 + 0 = 9;
- suma cifrelor numărului 99 este 9 + 9 = 18.

Clasa a VII-a

P070509: Ambalare

Într-un depozit au sosit mai multe materii prime, care trebuie ambalate. Se ştie că fiecare pachet poate conţine cel mult 10 kg de materie primă.

Fiecare materie primă trebuie ambalată astfel încât să se obţină un număr cât mai mic de pachete. Un pachet poate conţine o singură materie primă.

Se citesc de la tastatură numărul n (numărul materiilor prime) şi apoi numele şi cantitatea pentru fiecare dintre cele n materii prime.

Să se afișeze numărul maxim de pachete necesare pentru o anumită materie primă, precum și materiile prime care au necesitat acest număr de pachete.

Exemple

Intrare

```
7
Zinc 87
Fier 66
Argint 3
Aluminiu 98
Plumb 48
Carbune 83
Magneziu 16
```

Ieșire

```
10
Aluminiu
```

Intrare

```
8
Zinc 87
Fier 66
Argint 3
Aluminiu 88
Plumb 48
Carbune 83
Magneziu 16
Potasiu 13
```

Ieșire

```
9
Zinc
Aluminiu
Carbune
```

P070510: Text crescător

Se consideră un text format din mai multe cuvinte, separate între ele prin spații. În afara spațiilor, în text nu pot apărea decât literele mari și mici ale alfabetului englezesc. Textul va conține cel mult 200 de caractere.

Un text este numit *crescător*, dacă înlocuind fiecare cuvânt cu numărul său de litere, obținem un șir de numere ordonate crescător.

Să se afișeze pe ecran mesajul DA în cazul în care textul este crescător și mesajul NU în caz contrar.

Exemple

Intrare

```
Gazeta de Informatica
```

Ieșire

```
NU
```

Intrare

```
Ala Bala Portocala
```

Ieșire

```
DA
```

Intrare

```
unu doi trei
```

Ieșire

```
DA
```

P070511: Cifru

Pentru a-și proteja secretele atlanții folosesc un cod special, format din simboluri care sunt echivalente cifrelor arabe și literelor mici din alfabetul englezesc.

Pentru a putea accesa aceste secrete, trebuie urmată următoarea procedură:

- se identifică poziția primei cifre a codului;
- se identifică poziția ultimei cifre a codului;
- se determină diferența *d* dintre aceste poziții;
- se adaugă *d* la fiecare cifră a codului; dacă rezultatul este mai mare decât 10, se păstrează doar ultima cifră;
- se calculează suma *s* a cifrelor obținute;
- fiecare literă a codului este înlocuită cu a *s*-a literă care urmează după ea în alfabetul englezesc; se consideră că după litera z urmează litera a.

Literele obținute, scrise în ordine inversă, trebuie folosite în diversele dispozitive ale atlanților, pentru a descoperi secretele acestora.

Caracterele codului special vor fi citite de la tastatură, iar rezultatul va fi scris pe ecran.

Vor exista cel puțin două cifre și cel puțin două litere în codul citit.

Nu vor exista alte tipuri de caractere în afară de cifrele arabe și literele mici ale alfabetuli englezesc.

Codul va fi format din cel puțin 4 și din cel mult 200 de caractere.

Exemple

Intrare

```
2gazeta4de5informatica7
```

Ieșire

```
acitamrofniedatezag
```

Intrare

```
6ginfo6
```

Ieșire

```
sjrmk
```

Intrare

```
ginfo90
```

Ieșire

```
pgojh
```

Explicații

Pentru primul exemplu:
- prima cifră se află pe prima poziție;
- ultima cifră se află pe a douăzeci și treia poziție;
- diferența *d* dintre cele două poziții este 22;
- prin adăugarea valorii 22 la fiecare cifră și păstrarea doar a cifrei celei mai nesemnificative obținem:
  - cifra 5 devine 1 (5 + 22 = ~~24~~);
  - cifra 4 devine 6 (4 + 22 = ~~26~~);
  - cifra 5 devine 7 (4 + 22 = ~~27~~);
  - cifra 7 devine 9 (4 + 22 = ~~29~~);
- suma *s* a noilor cifre este 4 + 6 + 7 + 9 = 26;
- fiecare literă va fi înlocuită cu cea de-a douăzeci și șasea literă care urmează după ea; datorită faptului că alfabetul englezesc conține 26 de litere, ele vor rămâne aceleași;
- literele scrise în ordine inversă sunt

`acitamrofniedatezag.`

Pentru al doilea exemplu:
- prima cifră se află pe prima poziție;
- ultima cifră se află pe a șaptea poziție;
- diferența *d* dinre cele două poziții este 6;
- prin adăugarea valorii 6 la fiecare cifră și păstrarea doar a cifrei celei mai nesemnificative obținem:
  - cifra 6 devine 2 (6 + 6 = ~~12~~);
  - cifra 6 devine 2 (6 + 6 = ~~12~~);
- suma *s* a noilor cifre este 2 + 2 = 4;
- fiecare literă va fi înlocuită cu cea de-a patra literă care urmează după ea:
  - litera `g` devine litera `k`;
  - litera `i` devine litera `m`;
  - litera `n` devine litera `r`;
  - litera `f` devine litera `j`;
  - litera `o` devine litera `s`;
- literele scrise în ordine inversă sunt `sjrmk`.

Pentru al treilea exemplu:
- prima cifră se află pe a șasea poziție;
- ultima cifră se află pe a șaptea poziție;
- diferența *d* dinre cele două poziții este 1;
- prin adăugarea valorii 1 la fiecare cifră și păstrarea doar a cifrei celei mai nesemnificative obținem:
  - cifra 9 devine 0 (9 + 1 = ~~10~~);
  - cifra 0 devine 1 (0 + 1 = 1);
- suma *s* a noilor cifre este 0 + 1 = 1;
- fiecare literă va fi înlocuită cu litera care urmează după ea:
  - litera `g` devine litera `h`;
  - litera `i` devine litera `j`;
  - litera `n` devine litera `o`;
  - litera `f` devine litera `g`;
  - litera `o` devine litera `p`;
- literele scrise în ordine inversă sunt `pgojh`.

P070512: Șir pseudopalindrom

Se consideră un șir format din cel puțin două și cel mult 100 de caractere.

Numim ***permutare simplă*** a unui șir, șirul obținut prin eliminarea unui caracter de la începutul șirului și adăugarea sa la sfârșitul șirului.

Numim ***permutare*** a unui șir, șirul obținut prin aplicarea succesivă a mai multor permutări simple.

Numim un șir ***palindrom*** dacă prin oglindirea sa (citirea de la dreapta la stânga) se obține același șir.

Numim un șir ***pseudopalindrom*** dacă el sau una dintre permutările sale este un șir palindrom.

Să se afișeze pe ecran mesajul `DA` în cazul în care șirul este pseudopalindrom și mesajul `NU` în caz contrar.

Exemple

Intrare
`Gazeta de Informatica`

Ieșire
`NU`

Intrare
`rotomotor`

Ieșire
`DA`

Intrare
`aabb`

Ieșire
`DA`

Explicații
- niciuna dintre permutările șirului `Gazeta de Informatica` nu este un șir palindrom; se poate observa că, pentru ca un șir să fie pseudopalindrom, fiecare caracter, cu excepția cel mult a unuia, trebuie să apară în șir de un număr par de ori; pentru exemplul de față, caracterele `G`, `z`, `d`, `I`, `n`, `f`, `r`, `m`, `i` și `c` apar o singură dată; condiția menționată este ***necesară***, nu și *suficientă*;
- șirul `rotomotor` este el însuși palindrom, deci, potrivit definiției, este și pseudopalindrom;
- șirul `aabb` are două permutări (`abba` și `baab`) care reprezintă un șir palindrom.

Clasa a VIII-a

P070513: Numere puternice

Spunem că numărul natural nenul și neprim *a* este mai *puternic* decât numărul natural nenul și neprim *b* dacă mulțimea divizorilor proprii ai numărului *b* are cardinalul strict mai mic decât mulțimea divizorilor proprii ai numărului *a*. Să se scrie un program care citește o valoarea *n*, un vector *A* format din *n* ($1 < n < 50$) numere naturale nenule și neprime, o valoarea *m* și un vector *B* format din *m* ($1 < m < 50$) numere naturale nenule și neprime, iar apoi afișează, pe câte o linie, pentru fiecare număr din vectorul *A*,

numerele din vectorul *B* față de care este mai puternic. Dacă numărul nu este mai puternic decât niciunul dintre numerele din vectorul *B*, linia corespunzătoare va conține doar caracterul -.

Exemple

Intrare

```
3
12 81 54
4
6 54 20 10
```

Ieșire

```
6 10

6 20 10
```

Intrare

```
4
4 66 35 21
3
24 70 105
```

Ieșire

```
-
-
-
-
```

Explicații

Pentru primul exemplu avem:

- mulțimea divizorilor proprii ai numărului 12 este $dp(12) = \{2, 3, 4, 6\}$;
- mulțimea divizorilor proprii ai numărului 81 este $dp(81) = \{3, 9, 27\}$;
- mulțimea divizorilor proprii ai numărului 54 este $dp(54) = \{2, 3, 6, 9, 18, 27\}$;
- mulțimea divizorilor proprii ai numărului 6 este $dp(6) = \{2, 3\}$;
- mulțimea divizorilor proprii ai numărului 20 este $dp(20) = \{2, 4, 5, 10\}$;
- mulțimea divizorilor proprii ai numărului 10 este $dp(10) = \{2, 5\}$;
- numărul 12 are patru divizori proprii și este mai puternic decât numerele 6 și 10 care au câte doi divizori proprii;
- numărul 81 are trei divizori proprii și este mai puternic decât numerele 6 și 10 care au câte doi divizori proprii;
- numărul 54 are șase divizori și este mai puternic decât numerele 6, 20 și 10 care au doi, patru și, respectiv, doi divizori.

Pentru al doilea exemplu avem:

- mulțimea divizorilor proprii ai numărului 4 este $dp(4) = \{2\}$;
- mulțimea divizorilor proprii ai numărului 66 este $dp(66) = \{2, 3, 6, 11, 22, 33\}$;
- mulțimea divizorilor proprii ai numărului 35 este $dp(35) = \{5, 7\}$;
- mulțimea divizorilor proprii ai numărului 21 este $dp(21) = \{3, 7\}$;
- mulțimea divizorilor proprii ai numărului 24 este $dp(24) = \{2, 3, 4, 6, 8, 12\}$;
- mulțimea divizorilor proprii ai numărului 70 este $dp(70) = \{2, 5, 7, 10, 14, 35\}$;
- mulțimea divizorilor proprii ai numărului 105 este $dp(105) = \{3, 5, 7, 15, 21, 35\}$;
- numărul 4 are un singur divizor propriu și nu este mai puternic decât niciunul dintre numerele din al doilea șir;
- numărul 66 are șase divizori și nu este mai puternic decât niciunul dintre numerele din al doilea șir;
- numărul 35 are doi divizori și nu este mai puternic decât niciunul dintre numerele din al doilea șir;
- numărul 21 are doi divizori și nu este mai puternic decât niciunul dintre numerele din al doilea șir.

P070514: Numere impresionante

Spunem că un număr este *impresionant* dacă atât el, cât și oglinditul său (numărul citit de la stânga la dreapta), sunt numere prime.

Se citesc de la tastatură două numere întregi *a* și *b*, cu valori cuprinse între 1 și 1000000. La ieșire trebuie scrise, pe câte o linie, toate numerele impresionante din intervalul $[a, b]$. Dacă în intervalul respectiv nu există niciun număr impresionant, atunci la ieșire se va scrie o singură linie pe care se va afla doar caracterul -.

Exemple

Intrare

```
5 20
```

Ieșire

```
5
7
11
13
17
```

Intrare

```
2457 2999
```

Ieșire

```
-
```

Explicații

Pentru primul exemplu avem:

- numerele 5 și 7 sunt impresionante deoarece sunt formate dintr-o singură cifră (deci sunt identice cu oglinditele lor) și sunt prime;
- numărul 11 este impresionant deoarece este prim și palindrom (identic cu oglinditul său);
- numărul 13 este impresionant deoarece este prim, iar oglinditul său, 31, este și el prim;

- numărul 17 este impresionant deoarece este prim, iar oglinditul său, 71, este și el prim.

Pentru al doilea exemplu nu avem niciun număr impresionant deoarece oglinditele tuturor numerelor cuprinse între 2457 și 2999 sunt pare, deci neprime.

P070515: Rating

Rating-urile TV sunt măsurate cerând telespectatorilor să precizeze care au fost intervalele de timp în care au urmărit anumite posturi de televiziune.

Fiecare post primește apoi aceste intervale și poate determina momentele de maximă audiență.

Telespectatorii descriu intervalele prin minutul de început și minutul de sfârșit (ambele sunt valori cuprinse între 0 și 1440). Se consideră că minutul de început face parte din interval, iar cel de sfârșit nu face parte din interval. După primirea datelor, un anumit post de televiziune dorește să determine minutele în care audiența sa a fost maximă.

De la intratea standard se va citi un număr *n*, reprezentând numărul intervalelor, precum și *n* perechi de numere, reprezentând intervalele.

Minutele de maximă audiență vor fi scrise la ieșirea standard, câte unul pe o linie.

Exemple

Intrare

```
20
1000 1200
1005 1200
1010 1200
1015 1195
1010 1030
1020 1150
1075 1200
1025 1035
1030 1035
1025 1040
1070 1075
1010 1040
1030 1040
1000 1100
1070 1200
1030 1040
1250 1300
1275 1325
1010 1020
1400 1410
```

Ieșire

```
1030
1031
1032
1033
1034
```

Intrare

```
5
10 20
15 25
20 30
25 35
35 40
```

Ieșire

```
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
```

Explicații

Pentru primul exemplu, minutele 1030, 1031, 1032, 1033 și 1034 sunt cuprinse în 12 intervale:

- [1000, 1200);
- [1005, 1200);
- [1010, 1200);
- [1015, 1195);
- [1020, 1150);
- [1075, 1200);
- [1025, 1035);
- [1030, 1035);
- [1025, 1040);
- [1010, 1040);
- [1030, 1040);
- [1000, 1100).

Pentru a doilea exemplu, minutele cuprinse între minutul 15 și minutul 34 (inclusiv) apar în câte două intervale:

- minutele cuprinse între minutul 15 și minutul 19 (inclusiv) apar în intervalele [10, 20) și [15, 25);
- minutele cuprinse între minutul 20 și minutul 24 (inclusiv) apar în intervalele [15, 25) și [20, 30);
- minutele cuprinse între minutul 25 și minutul 29 (inclusiv) apar în intervalele [20, 30) și [25, 35);
- minutele cuprinse între minutul 30 și minutul 34 (inclusiv) apar în intervalele [25, 35) și [30, 40).

P070516: Decriptare

Trebuie să spargem un cod de criptare! Din fericire avem la dispoziție următoarele infromații:

- textul este format doar din litere mici ale alfabetului englezesc;
- fiecare literă a fost înlocuită cu ce-a de-a *k*-a literă care urmează în alfabet după ea; dacă se ajunge la litera z, se va continua de la litera a (valoarea *k* este necunoscută);
- cea mai frecvent utilizată literă în textul care va fi decriptat este a.

Scrieți un program care citește de la tastatură un text criptat format din cel mult 100 de litere și afișează pe ecran textul decriptat.

Exemple

Intrare

```
icbgvcfgkphqtocvkec
```

Ieșire

```
gazetadeinformatica
```

Intrare

```
xxxxxx
```

Ieșire

```
aaaaaa
```

Intrare

```
abbba
```

Ieșire

```
zaaaz
```

Clasa a IX-a

P070517: Spirală

Se consideră primele n^2 numere ale unui șir al cărui prim termen este numărul 1 și fiecare nou termen este cel mai mic număr prim strict mai mare decât dublul termenului anterior. Așadar, primele elemente ale șirului sunt 1, 3, 7, 17, 37, ...

Cele n^2 numere trebuie aranjate *în spirală*, într-o matrice pătratică de dimensiune *n*.

Aranjarea în spirală începe amplasând numerele, în ordine, pe prima linie până se ajunge la ultima coloană; continuăm în jos pe ultima coloană până ajungem la ultima linie; pe această linie ne deplasăm spre stânga până ajungem pe prima coloană; urcăm apoi până pe a doua linie; completăm în continuare a doua linie, penultima coloană, penultima linie, a doua coloană, a treia linie și așa mai departe, până în centrul spiralei.

Date de intrare

Fișierul de intrare **SPIRALA.IN** conține pe prima linie numărul natural.

Date de ieșire

Fișierul de ieșire **SPIRALA.OUT** va conține *n* linii, pe fiecare dintre acestea aflându-se câte *n* numere care vor reprezenta elementele matricei care conține spirala.

Restricție

- $1 < n \leq 10$.

Exemplu

SPIRALA.IN

```
3
```

SPIRALA.OUT

```
1 3 7
331 673 17
163 79 37
```

Explicație

- cel mai mic număr prim strict mai mare decât $2 \cdot 1 = 2$ este 3;
- cel mai mic număr prim strict mai mare decât $2 \cdot 3 = 6$ este 7;
- cel mai mie număr prim strict mai mare decât $2 \cdot 7 = 14$ este 17;
- cel mai mic număr prim strict mai mare decât $2 \cdot 17 = 34$ este 37;
- cel mai mic număr prim strict mai mare decât $2 \cdot 37 = 74$ este 79;
- cel mai mic număr prim strict mai mare decât $2 \cdot 79 = 158$ este 163;
- cel mai mic număr prim strict mai mare decât $2 \cdot 163 = 326$ este 331;
- cel mai mic număr prim strict mai mare decât $331 \cdot 2 = 662$ este 673.

Timp de execuție: o secundă/test

P070518: Numere romane extinse

Romanii foloseau doar simbolurile I, V, X, L, C, D și M pentru a construi numeralele. Valorile corespunzătoare acestor cifre sunt:

- I - 1 (unu - unus);
- V - 5 (cinci - quinque);
- X - 10 (zece - decem);
- L - 50 (cincizeci - quinquaginta);
- C - 100 (o sută - centum);
- D - 500 (cinci sute - quingenti);
- M - 1000 (o mie - mille).

Simbolurile erau alăturate (în principiu, cele cu valori mari la început) pentru a construi numeralele. De exemplu, numărul 2007 este reprezentat prin MMVII.

De asemenea, puteau fi folosite anumite prescurtări prin plasarea unui simbol cu valoare mai mică (cu rol de scădere) înaintea unui simbol cu o valoare mai mare. Astfel, numărul 2457 s-ar scrie MMCDLVII. Totuși, nu este

permisă orice combinație de simboluri. Un simbol cu valoare mai mică poate apărea înaintea unuia cu valoare mai mare doar în combinațiile IV, IX, XL, XC, CD sau CM. Nu sunt permise combinații de tipul IC, IIC sau IM. Așadar, pentru 98 vom scrie XCVIII, pentru 99 vom scrie XCIX, iar pentru 999 vom scrie CMXCIX.

Din nefericire, pentru numerele de la 3000 în sus, scrierea ar deveni prea greoaie folosind doar aceste simboluri. Pentru a se evita lungi șiruri de M - uri, s-a introdus o bară deasupra unui simbol pentru a indica înmulțirea cu 1000.

Se foloseau următoarele simboluri:

- $\overline{V}$ - 5 (cinci mii);
- $\overline{X}$ - 10 (zece mii);
- $\overline{L}$ - 50 (cincizeci de mii);
- $\overline{C}$ - 100 (o sută de mii);
- $\overline{D}$ - 500 (cinci sute de mii);
- $\overline{M}$ - 1000 (un milion).

Dacă era nevoie de numere mai mari, se foloseau două bare deasupra sau una sub simbol. Simbolurile cu bare dedesub sunt:

- $\underline{V}$ - 5 (cinci milioane);
- $\underline{X}$ - 10 (zece milioane);
- $\underline{L}$ - 50 (cincizeci de milioane);
- $\underline{C}$ - 100 (o sută de milioane);
- $\underline{D}$ - 500 (cinci sute de milioane);
- $\underline{M}$ - 1000 (un miliard).

Practic, folosind aceste simboluri, am putea reprezenta relativ compact toate numerele întregi cu semn reprezentate pe 32 de biți.

Va trebui să scrieți un program care transformă un număr scris cu cifre romane, folosind toate simbolurile și regulile prezentate, în numărul echivalent scris cu cifre arabe.

Pentru a reprezenta simbolurile cu valori mai mari decât o mie vom folosi litera corespunzătoare, urmată de una sau două liniuțe de subliniere.

Așadar, simbolurile folosite pentru scrierea numărului ar putea fi I, V, X, L, C, D, M, V_, X_, L_, C_, D_, M_, V__, X__, L__, C__, D__ și M__.

Date de intrare

Fișierul de intrare **ROMAN.IN** conține pe prima linie un număr valid scris cu cifre romane.

Date de ieșire

Fișierul de ieșire **ROMAN.OUT** va conține pe prima linie numărul din fișierul de intrare, scris cu cifre romane.

Restricție

- numărul care trebuie transformat este cuprins între 1 și 2000000000.

Exemple

ROMAN.IN

```
MMCDLVII
```

ROMAN.OUT

```
2457
```

ROMAN.IN

```
M__I
```

ROMAN.OUT

```
1000000001
```

ROMAN.IN

```
M__C__M__X__C__M_X__M_C_M_X_C_MX_MCMXCIX
```

ROMAN.OUT

```
1999999999
```

Timp de execuție: o secundă/test

P070519: Taxa de protecție

Șeful bandei din cartier trebuie să parcurgă lunar un traseu format din n locuințe pentru a încasa diverse taxe de protecție, dar și a plăti diverse comisioane. El parcurge locuințele începând de la prima și terminând cu cea de-a n-a, vizitând fiecare locuință o singură dată.

Se știe că șeful bandei va primi bani cel puțin în una dintre locuințe. Mai mult, se cunosc exact sumele care trebuie cerute sau plătite.

În urma pierderii unui pariu legat de rezultatul unei obscure partide de fotbal, șeful a fost nevoit să se împrumute de la un cămătar și acum a sosit momentul să restituie împrumutul. Pentru a rămâne cu cât mai mulți bani, șeful va alege două locuințe l_i și l_j de pe traseu și va parcurge traseul dintre ele. Evident, va face alegerea în așa fel încât, după efectuarea plăților și încasarea taxelor, suma pe care o are la dispoziție să fie maximă.

Șeful bandei va trece prin toate locuințele de pe traseu, chiar dacă în unele va trebui să plătească diverse comisioane. Pentru a nu amplifica inutil tensiunea din cartier datorată așteptării (locuințele nevizitate acum vor fi vizitate cu întârziere), dacă există mai multe variante care duc la obținerea unei aceleiași sume maxime, atunci este preferată cea în care sunt vizitate mai multe locuințe.

Scrieți un program care determină suma maximă posibilă care poate fi strânsă de șeful bandei, precum și secvența cea mai lungă de locuințe care permite obținerea sumei respective. În cazul în care o aceeași sumă maximă poate fi obținută pentru alegeri diferite ale locuințelor l_i și l_j, vizitând același număr de locuințe, atunci va fi aleasă una dintre acestea și anume cea pentru care numărul de ordine al locuinței l_i este cel mai mic.

Date de intrare

Fișierul de intrare **TAXA.IN** conține pe prima linie numărul natural n, reprezentând numărul locuințelor de pe traseul inițial.

Pe următoarea linie se află n numere întregi, separate prin câte un spațiu, reprezentând sumele pe care șeful ban-

dei ar trebui să le încaseze sau să le plătească în fiecare locuință; valorile pozitive reprezintă încasări, în timp ce cele negative reprezintă plăți.

Date de ieșire

Fișierul de ieșire **TAXA.OUT** va conține pe prima linie un număr natural nenul, care va reprezenta suma maximă care poate fi strânsă de șeful bandei.

Pe cea de-a doua linie vor fi scrise două numere naturale cuprinse între 1 și n, reprezentând numerele de ordine ale locuințelor care delimitează traseul ales.

Restricții și precizări

- $1 < n \leq 1000000$;
- sumele care trebuie încasate sau plătite sunt cuprinse între 1 și 1000000.
- suma maximă care poate fi strânsă nu va depăși valoarea de 1000000000.

Exemplu

TAXA.IN

```
10
4 -7 6 7 -2 2 -1 -2 2 -7
```

SPIRALA.OUT

```
13
3 6
```

Explicații

Pentru traseul care începe de la a treia și se încheie la a șasea locuință, suma strânsă este 6 + 7 - 2 + 2 = 13.

Aceeași sumă poate fi strânsă alegând traseul care începe de la a treia și se încheie la a patra locuință, dar acest traseu este mai scurt.

Timp de execuție: 5 secunde/test

Clasa a X-a

P070520: Meteoriți

Din nefericire, o mulțime de meteoriți a căzut pe terenul pe care urma să se construiască prima clădire a noii colonii umane de pe *Marte*.

Terenul poate fi privit ca fiind o zonă dreptunghiulară. Punctele în care au căzut meteoriții au coordonate întregi.

Coloniștii trebuie să construiască acum o altă clădire, după un nou plan, fără ca aceasta să ocupe locuri atinse de meteoriți.

Clădirea construită trebuie să fie dreptunghiulară, să aibă laturile paralele cu axele de coordonate, iar coloniștii doresc ca ea să aibă o suprafață cât mai mare.

Date de intrare

Fișierul de intrare **METEOR.IN** conține pe prima linie coordonatele terenului (ordonata și abscisa colțului stânga-sus, respectiv ordonata și abscisa colțului dreapta-jos).

Pe următoarea linie se află numărul n al meteoriților care au căzut pe terenul respectiv.

Fiecare dintre următoarele n linii conține câte două numere întregi, reprezentând coordonatele la care a căzut unul dintre meteoriți.

Date de ieșire

Fișierul de ieșire **METEOR.OUT** va conține pe prima linie suprafață maximă a clădirii care poate fi construită.

Pe cea de-a doua linie se vor afla patru numere, reprezentând coordonatele (ordonata și abscisa) colțului din stânga-sus, respectiv ale colțului din dreapta-jos a clădirii.

Restricții și precizări

- toate coordonatele sunt numere naturale;
- $0 \leq n \leq 200$;
- dacă există mai multe soluții, va fi aleasă doar una dintre ele (oricare);
- zonele în care au căzut meteoriții sunt considerate a fi punctiforme, pot atinge marginile terenului sau marginile clădirii.

Exemplu

METEOR.IN

```
0 0 100 100
4
30 40
60 60
30 70
20 20
```

METEOR.OUT

```
4200
30 0 100 60
```

Timp de execuție: o secundă/test

P070521: Acorduri de pace

Într-o galaxie îndepărtată există n planete locuite. Între unele dintre civilizațiile care trăiesc pe aceste planete au fost semnate acorduri de pace.

Un acord de pace între două civilizații duce la următoarele obligații:

- cele două civilizații nu se pot ataca între ele;
- cele două civilizațiile devin aliate;
- niciuna dintre cele două civilizații nu poate ataca nicio civilizație aliată celeilalte civilizații;
- fiecare dintre cele două civilizații devine aliată a tuturor civilizațiilor aliate celeilalte civilizații.

Astfel, civilizațiile aliate între ele, direct sau indirect, formează confederații galactice.

Cunoscând numărul planetelor locuite, precum și lista acordurilor semnate, să se determine numărul confederațiilor din galaxie. De asemenea, trebuie determinate perechile de civilizații care ar trebui să semneze acorduri de

pace pentru ca în galaxie să rămână o singură confederație galactică. Numărul acestor acorduri trebuie să fie minim.

Date de intrare

Fișierul de intrare **GALACTIC.IN** conține pe prima linie un număr natural n, reprezentând numărul planetelor locuite și un număr natural m, reprezentând numărul acordurilor semnate.

Pe fiecare dintre următoarele m linii se află câte două numere care reprezintă numerele de ordine ale civilizațiilor între care există acorduri directe de pace.

Date de ieșire

Pe prima linie a fișierului de ieșire **GALACTIC.OUT** se va afla numărul k al confederațiilor galactice.

Pe următoarele k linii se vor afla numerele de ordine ale civilizațiilor care formează fiecare confederație galactică.

Dacă în galaxie există o singură confederație galactică, fișierul de ieșire se încheie.

Dacă în galaxie există mai multe condeferații galactice, atunci pe linia următoare va fi scris numărul minim t al acordurilor necesare pentru a se ajunge la o singură confederație.

Fiecare dintre următoarele t linii va conține două numere care reprezintă civilizații între care trebuie semnat un acord.

Restricții și precizări

- $1 \le n \le 100$;
- datele de intrare sunt corecte; fiecare acord este descris o singură dată;
- dacă există mai multe variante prin care se poate ajunge la o singură confederație galactică, poate fi aleasă oricare dintre ele;
- fiecare acord nou trebuie să ducă la unirea a două confederații care nu au fost unite prin niciunul dintre acordurile descrise anterior.

Exemple

GALACTIC.IN

```
8 6
1 2
1 3
1 4
2 4
5 8
6 7
```

GALACTIC.OUT

```
3
1 2 3 4
5 8
6 7
2
1 5
1 7
```

GALACTIC.IN

```
4 4
1 2
1 3
2 4
3 4
```

GALACTIC.OUT

```
1
1 2 3 4
```

Timp de execuție: o secundă/test

P070522: Litere

Se consideră o matrice ale cărei elemente sunt litere ale alfabetului englezesc. Spunem că un cuvânt apare în matrice dacă prima sa literă este un element al matricei, cea de-a doua literă este un element al matricei vecin primului (pe orizontală, verticală sau diagonală), cea de-a treia literă este un element vecin celui de-al doilea (dar diferit de primul). În general, cea de-a k literă trebuie să fie un element vecin celui care corespunde celei de-a k - 1 litere, dar diferit de oricare dintre elementele corespunzătoare primelor k - 2 litere.

Va trebui să determinați numărul total al aparițiilor unui cuvânt în matrice.

Date de intrare

Pe prima linie a fișierului de intrare **LITERE.IN** se află două numere naturale n și m, reprezentând dimensiunile matricei de litere.

Pe fiecare dintre următoarele n linii se află câte m caractere, care reprezintă câte o linie din matrice.

Pe ultima linie se va afla cuvântul al cărui număr de apariții trebuie determinat.

Date de ieșire

Fișierul de ieșire **METEOR.OUT** va conține pe prima linie numărul total al aparițiilor cuvântului dat în cadrul matricei de litere.

Restricție

- $1 \le m, n \le 255$.

Exemplu

LITERE.IN

```
5 5
gaoao
aiafa
oanao
afafa
oaoao
ginfo
```

LITERE.OUT

```
9
```

Explicații

```
gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao

gaoao
aiafa
oanao
afafa
oaoao
```

Timp de execuție: o secundă/test

P070523: Rute comerciale

Într-o galaxie îndepărtată există N planete care sunt centre comerciale. Cele N planete sunt legate între ele printr-un număr total de M găuri de vierme, prin care se poate circula în ambele sensuri.

Ghilda Galactică a Comercianților dorește să organizeze un serviciu de patrulare, pentru a limita pagubele cauzate de atacurile piraților spațiului.

Pentru aceasta trebuie stabilite traseele navetelor de patrulare, astfel încât să fie respectate următoarele condiții:

- un traseu este format din planete (cel puțin trei);
- două planete aflate pe poziții consecutive într-un traseu trebuie să fie legate direct printr-o gaură de vierme;
- prima și ultima planetă a traseului trebuie să fie legate între ele printr-o gaură de vierme.

Se dorește crearea unui număr maxim de astfel de trasee, astfel încât pentru parcurgerea fiecăruia dintre trasee să fie necesară utilizarea unei găuri de vierme care nu este utilizată de niciunul dintre celelalte trasee.

Date de intrare

Prima linie a fișierului de intrare **TRASEE.IN** conține numărul N al planetelor și numărul M al găurilor de vierme.

Fiecare dintre următoarele M linii conține câte două numere x și y cu semnificația: între planetele x și y există o gaură de vierme.

Date de ieșire

Fișierul de ieșire **TRASEE.OUT** va conține pe prima linie numărul maxim K al traseelor care pot fi stabilite.

Pe fiecare dintre următoarele K linii va fi descris câte un traseu. Primul număr de pe o astfel de linie indică numărul p al planetelor care formează traseul. Următoarele p numere de pe linia respectivă reprezintă planetele care formează traseul, în ordinea în care trebuie parcurse acestea.

Restricții și precizări

- $3 \le N \le 100$;
- $1 \le M \le 1000$;
- există cel mult o gaură de vierme între oricare două planete din galaxie;
- traseele pot fi descrise în orice ordine;
- dacă există mai multe soluții, trebuie generată, la alegere, doar una dintre ele.

Exemplu

TRASEE.IN

```
10 13
1 2
1 3
2 3
2 4
```

```
3 4
3 5
4 8
5 6
5 7
6 7
8 9
8 10
9 10
```

TRASEE.OUT

```
4
3 1 2 3
4 1 2 4 3
3 5 6 7
3 8 9 10
```

Timp de execuție: o secundă/test

P070524: Spioni sardaukari

Serviciul de Informații al *Împaratului Padișah Shaddam Corrino al IV-lea*, împărat a peste un milion de lumi, este format din N sardaukari special antrenați.

Din motive de securitate, majoritatea spionilor sardaukari nu pot comunica direct unul cu altul. Totuși, rețeaua de spionaj este creată în așa fel încât, prin intermediari, un spion să poată trimite informații oricărui alt spion.

Configurația curentă a serviciului de informații este considerată a fi sigură în cazul în care, prin eliminarea unui sardaukar, nu este afectată comunicarea dintre ceilalți agenți.

În primul rând, *Contele Hasimir Fenring* dorește să știe dacă actuala configurație este sigură. În cazul în care configurația nu este sigură, atunci el dorește să știe care sunt sardaukarii vulnerabili, care sunt legăturile vulnerabile și care sunt grupurile de sardaukari care ar putea forma configurații sigure maximale.

Un sardaukar este vulnerabil dacă, prin eliminarea sa, va apărea cel puțin o pereche de spioni sardaukari, care nu vor mai putea comunica între ei.

O legătură este vulnerabilă dacă, prin compromiterea sa, va apărea cel puțin o pereche de spioni sardaukari, care nu vor mai putea comunica între ei.

Date de intrare

Fișierul de intrare **SHADDAM.IN** conține pe prima linie un număr natural n, reprezentând numărul total al spionilor sardaukari, precum și un număr natural m, reprezentând numărul total al perechilor de sardaukari care pot comunica direct între ei.

Fiecare dintre următoarele m linii conține câte două numere x și y cu semnificația: spionii x și y pot comunica direct între ei.

Date de ieșire

Prima linie a fișierului de ieșire **SHADDAM.OUT** va conține mesajul DA în cazul în care serviciul de informații are o configurație sigură, respectiv mesajul NU, dacă actuala configurație nu este sigură.

Cea de-a doua linie a fișierului va conține numărul p al spionilor vulnerabili.

Fiecare dintre următoarele p linii va conține numărul de ordine ale unuia dintre spionii vulnerabili.

Următoarea linie a fișierului va conține numărul q al legăturilor vulnerabile dintre sardaukari.

Fiecare dintre următoarele q linii va conține câte două numere x și y cu semnificația: legătura directă dintre spionii x și y este vulnerabilă.

Următoarea linie a fișierului va conține numărul r al grupurilor maximale de sardaukari care ar putea forma configurații sigure.

Fiecare dintre următoarele r linii va conține numerele de ordine ale sardaukarilor care formează grupul corespunzător.

Restricții și precizări

- $1 \le N < 100$;
- $1 \le M < 1000$;
- în cazul în care configurația curentă este sigură, nu vom avea niciun spion vulnerabil, nicio legătură vulnerabilă și un singur grup de spioni, format din toți sardaukarii;
- fiecare dintre spionii vulnerabili va apărea o singură dată în listă;
- fiecare legătură vulnerabilă trebuie descrisă o singură dată;
- grupurile care constituie configurații sigure pot fi descrise în orice ordine;
- spionii care formează un grup pot apărea în orice ordine pe linia corespunzătoare.

Exemplu

SHADDAM.IN

```
10 13
1 2
1 3
2 3
2 4
3 4
3 5
4 8
5 6
5 7
6 7
8 9
8 10
9 10
```

SHADDAM.OUT

```
NU
4
3
4
5
8
```

```
2
3 5
4 8
5
1 2 3 4
3 5
4 8
5 6 7
8 9 10
```

SHADDAM.IN

```
4 5
1 2
1 3
2 3
2 4
3 4
```

SHADDAM.OUT

```
DA
0
0
1
1 2 3 4
```

Timp de execuție: o secundă/test

P070525: Copaci

Alaron, regele elfilor, organizează un concurs interesant. El a marcat $2 \cdot N$ copaci în padure și le-a cerut celor N participanți să aleagă, pe rând, perechi de copaci.

Punctajul obținut de fiecare elf va fi direct proporțional cu distanța dintre copacii aleși.

Evident, fiecare elf va dori să obțină punctajul cel mai mare, deci va alege pereche de copaci aflați la cea mai mare distanță în momentul alegerii.

Pentru fiecare dintre cei $2 \cdot N$ copaci se cunosc coordonatele la care se află, iar dimensiunile trunchiurilor lor sunt neglijabile.

Date de intrare

Fișierul de intrare **ALARON.IN** conține pe prima linie un număr natural n, reprezentând numărul total al elfilor care participă la concurs.

Fiecare dintre următoarele $2 \cdot N$ linii conține coordonatele unuia dintre copacii aleși de *Regele Alaron*.

Date de ieșire

Prima linie a fișierului de ieșire **ALARON.OUT** va conține N linii, pe fiecare dintre acestea aflându-se trei numere.

Primul număr va reprezenta distanța dintre copacii aleși de elful corespunzător, iar celelalte două numere vor reprezenta numerele de ordine ale copacilor aleși.

Distanțele vor fi scrise în ordinea alegerilor. Cu alte cuvinte, pe prima linie se vor afla distanța dintre copacii aleși de primul elf (și numerele de ordine ale acestora), pe a doua linie se va afla distanța maximă dintre doi dintre copacii rămași după eliminarea primilor doi și așa mai departe.

Restricții și precizări

- $1 \le N < 500$;
- $1 \le M < 1000$;
- nu pot exista doi sau mai mulți copaci la aceleași coordonate;
- toate coordonatele sunt numere întregi cuprinse între 0 și 1000;
- dacă un elf are mai multe opțiuni de alegere a celor doi copaci, atunci el poate alege oricare dintre opțiunile respective;
- distanțele din fișierul de ieșire vor fi scrise cu două zecimale exacte.

Exemplu

ALARON.IN

```
4
0 0
10 0
20 0
0 10
20 10
0 20
10 20
20 20
```

ALARON.OUT

```
2.83 1 8
2.83 3 6
2.00 2 7
2.00 4 5
```

ALARON.IN

```
5
0 0
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
0 9
```

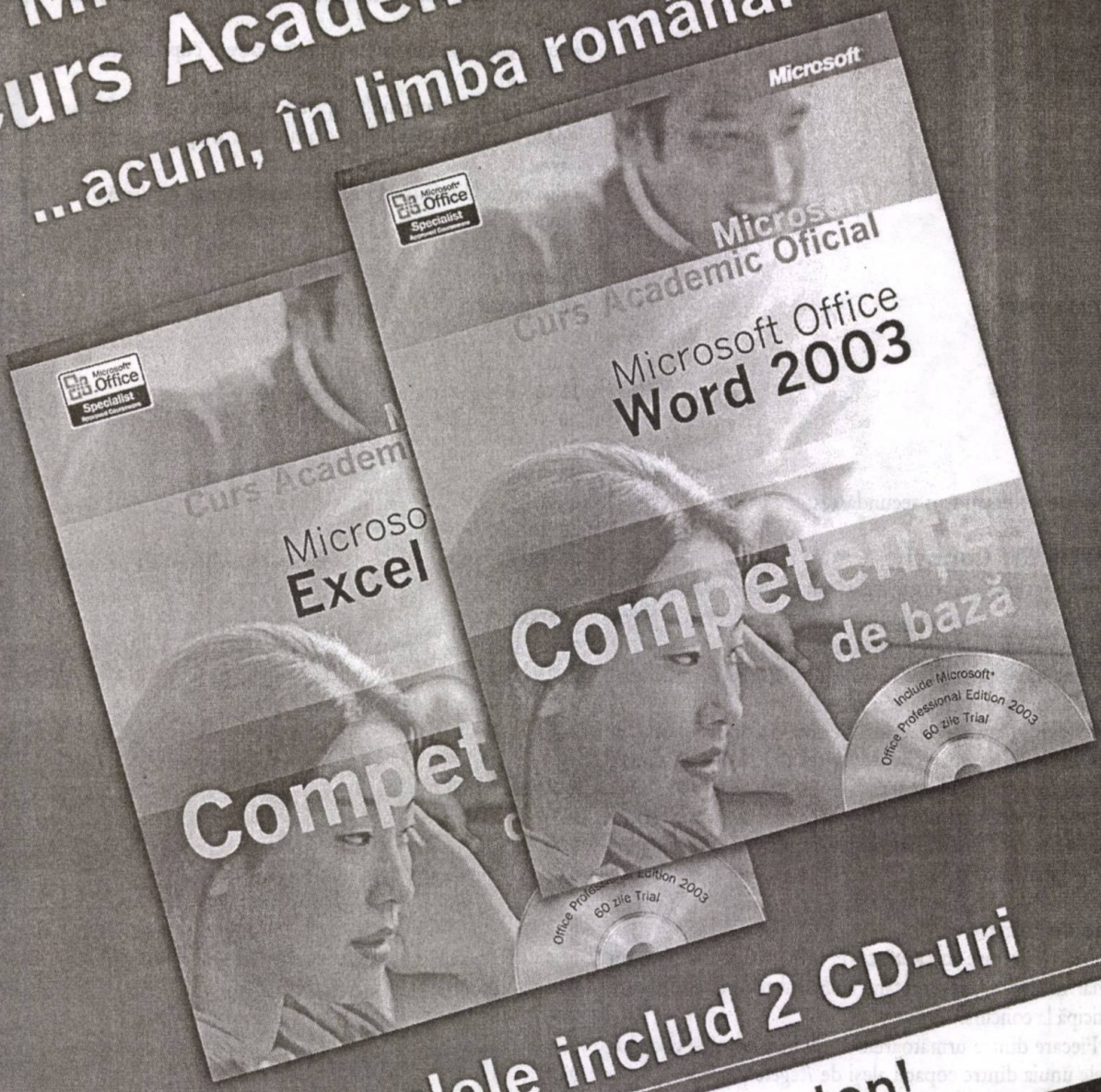
ALARON.OUT

```
9.00 1 10
7.00 2 9
5.00 3 8
3.00 4 7
1.00 5 6
```

Timp de execuție: o secundă/test

NOU!

Microsoft®
Curs Academic Oficial
...acum, în limba română!



Manualele includ 2 CD-uri

Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro

BYBLOS

Asupra unei PROBLEME de la ONI

Ion Petre

În cadrul acestui articol vom prezenta trei soluții pentru o problemă propusă spre rezolvare la ediția din 2001 a Olimpiadei Naționale de Informatică.

Enunțul problemei

Se consideră un dreptunghi cu lungimea egală cu $2 \cdot n$ centimetri și lățimea egală cu trei centimetri. Să se determine numărul m al pavărilor distincte care pot fi obținute folosind dale dreptunghiulare care au toate lungimea egală cu un centimetru și lățimea egală cu doi centimetri.

Date de intrare

Datele din fișierul de intrare **pavari.in** sunt structurate astfel:

- pe prima (și singura) linie se află numărul natural nenul n, reprezentând jumătatea lungimii dreptunghiului.

Date de ieșire

Datele din fișierul text de ieșire **pavari.out** trebuie structurate astfel:

- pe prima (și singura) linie se va scrie numărul natural nenul m, reprezentând numărul modalităților de a pava dreptunghiul.

Restricție

- $1 \le n \le 100$.

Exemplu

pavari.in

2

pavari.out

11

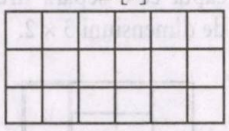
Timp de execuție: o secundă / test

Indicații

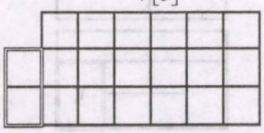
Vom defini următoarele suprafețe:

- $U[k]$ - dreptunghi cu lungimea egală cu $2 \cdot k$ și lățimea egală cu 3;
- $V[k]$ - dreptunghiul $U[k]$ căruia i se atașează o dală de dimensiuni 2 × 1.

$U[3]$



$V[3]$

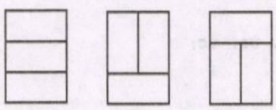


Problema tăieturilor în spațiu

Numărul pavărilor pentru o suprafață cu lungimea egală cu $2 \cdot k$ (U_k) depinde de:

- numărul pavărilor pentru o suprafață cu lungimea egală cu $2 \cdot (k - 1)$ (U_{k-1});
- numărul pavărilor (V_{k-2}) pentru o suprafață cu:
  - o lungime egală cu $2 \cdot (k - 2)$;
  - o lungime egală cu $2 \cdot (k - 2) + 1$;
  - lățimea egală cu 3.

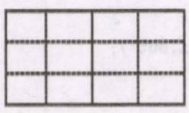
Pentru $n = 1$, numărul pavărilor este $U_1 = 3$:



Pentru $n = 2$, numărul pavărilor este : $U_2 = 3 \cdot U_1 + 2 \cdot V_0$. Avem $V_0 = 1$, deoarece suprafața $V[0]$ constă într-o singură dală, de dimensiuni 2 × 1.

Explicații

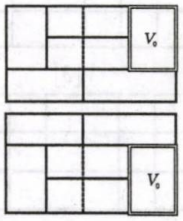
- numărul pavărilor unei suprafețe de dimensiuni 3 × 2 este $U_1 = 3$;



Listing PAVARI1.CPP

```
#include<fstream.h>
#include<math.h>

unsigned long vc, va, ua, uc;
int n, k;

void main(void)
{
  ifstream f("pavari.in");
  f >> n;
  f.close();

  va = 1;
  ua = 3;
  for (k = 2; k <= n; k++)
  {
    vc = ua + va;
    uc = 3 * ua + 2 * va;
    va = vc;
    ua = uc;
  }

  ofstream g("pavari.out");
  g >> uc;
  g.close();
}
```

- 2 = Numărul pavărilor unei suprefețe de dimensiuni 3 × 4 care taie dreapta care separă dreptunghiul în două dreptunghiuri de dimensiuni 3 × 2.



Pentru $n \geq 2$, relațiile de recurență obținute sunt următoarele:

- $U_{n+1} = 3 \cdot U_n + 2 \cdot V_{n-1}$;
- $V_n = U_n + V_{n-1}$.

Prima soluție

Pe baza formulelor prezentate mai sus, se obține programul **PAVARI1.CPP**, al cărui cod sursă este redat în casetă.

Listing PAVARI2.CPP

```
#include<fstream.h>
#include<math.h>

unsigned long uaa, ua, uc;
int n, k;

void main(void)
{
  ifstream f("pavari.in");
  f >> n;
  f.close();

  uaa=3;
  ua=11;
  for (k = 3; k <= n; k++)
  {
    uc = 4 * ua - uaa;
    uaa = ua;
    ua = uc;
  }

  ofstream g("pavari.out");
  g>>uc;
  g.close();
}
```

Listing PAVARI3.CPP

```
#include<fstream.h>
#include<math.h>

unsigned long uc;
int n, i, k;

void main(void)
{
  ifstream f("pavari.in");
  f >> n;
  f.close();

  uc = ceil((pow(3, 0.5) + 1) / 2 / pow(3, 0.5) *
                                  pow(2 + pow(3, 0.5), n) +
                          (pow(3, 0.5) - 1)/ 2 / pow(3, 0.5) *
                                  pow(2 - pow(3, 0.5), n));

  ofstream g("pavari.out");
  g<<uc;
  g.close();
}
```

A doua soluție

Dacă exprimăm, pe baza celei de-a doua relații de recurență, $V_{n-1} = U_{n-1} + V_{n-2}$, iar apoi înlocuim valoarea V_{n-1} în prima relație de recurență, vom obține:

$$U_{n+1} = 3 \cdot U_n + 2 \cdot U_{n-1} + 2 \cdot V_{n-2}.$$

Pe baza primei relații de recurență obținem:

$$2 \cdot V_{n-2} = U_n - 3 \cdot U_{n-1}.$$

Ajungem astfel la relația:

$$U_{n+1} = 4 \cdot U_n - U_{n-1}.$$

Va trebui să cunoaștem valorile $U_1 = 3$ și $U_2 = 11$ pentru a putea determina celelalte valori.

Ajungem astfel la programul **PAVARI2.CPP**, al cărui cod sursă este redat în casetă.

A treia soluție

Se determină expresia termenului general a relației de recurență. Avem $R^2 - 4 \cdot R + 1 = 0$ (ecuația caracteristică a relației de recurență); obținem $R_1 = 2 + 3^{1/2}$ și $R_1 = 2 - 3^{1/2}$.

Termenul general va fi:

$$U_k = c_1 \cdot (2 + 3^{1/2})^k + c_2 \cdot (2 + 3^{1/2})^k.$$

Valorile c_1 și c_2 pot fi determinate folosind relațiile $U_1 = 3$ și $U_2 = 11$. Obținem astfel termenul general ca fiind:

$$U_k = (1 + 3^{1/2}) / (2 \cdot 3^{1/2}) \cdot (2 + 3^{1/2})^k + (3^{1/2} - 1) / (2 \cdot 3^{1/2}) \cdot (2 + 3^{1/2})^k.$$

Programul **PAVARI3.CPP** reprezintă implementarea acestei soluții; timpul de rulare este de sub 0.01 secunde.

Bibliografie

1. Ioan Tomescu, *Probleme de combinatorică și teoria grafurilor*, Editura Didactică și Pedagocică, București;
2. Petre Ion, *Teorie și aplicații practice de laborator*, Editura Agora, Craiova;
3. Cătălin-Petru Nicolescu, *Sinteze de matematică*, Editura Albatros, București.

Cadrele didactice se implică în proiectul

Intel®Teach

Parte a inițiativei Intel®Education, Intel®Teach este un program educațional global, lansat în anul 2000 și prezent deja în 40 de țări. Implementarea programului Intel®Teach în școlile din România a debutat în ianuarie 2007, și face parte din amplul program național de informatizare a învățământului preuniversitar -SEI. Unul dintre cele mai apreciate proiecte de eGovernment din Europa, SEI asigură accesul elevilor și profesorilor din peste 4800 de licee și școli din România, la resurse informatice pentru instruirea asistată de calculator. Proiectul Intel®Teach utilizează platforma AeL, deoarece aceasta este folosită în majoritatea școlilor din țară și face parte din Strategia Europeană de eLearning elaborată de Comisia Europeană. Partenerul local pentru implementarea acestui proiect în școlile din România este SIVECO Romania.

În cadrul diverselor evenimente cu tematică educațională, profesorii s-au dovedit a fi extrem de receptivi și entuziaști în ceea ce privește implicarea în acest proiect. Îmbunătățirea continuă a procesului educațional reprezintă un obiectiv prioritar, iar accesul la acest program va facilita cadrelor didactice dezvoltarea abilităților de predare cu ajutorul instrumentelor informatice precum și obținerea unei acreditări suplimentare.

Oportunitatea de a participa la sesiunile de instruire Intel®Teach, vine în sprijinul profesorilor români, ajutându-i să integreze instrumentele și resursele tehnologice în procesul de educație, printr-un program regulat de ateliere de lucru și seminarii axate pe metode moderne de educație și management școlar.

În următorii trei ani, peste 60.000 de profesori români vor beneficia de training în cadrul acestui program. În urma instruirii, dascălii vor învăța cum să alinieze programa școlară la standardele moderne și cum să utilizeze instrumentele IT pentru a putea evalua mai bine activitatea elevilor. Acest proiect dorește să evidențieze importanța utilizării computerelor și internetului ca instrumente de cercetare, comunicare și dezvoltare a creativității.

Sunt sceptici care consideră că programele de formare profesională adresate profesorilor sunt inutile, dar programul internațional global Intel®Teach și-a demonstrat eficiența, instruind până în prezent peste 4 milioane de profesori în lume. Rezultatele cercetărilor demonstrează că noile tehnologii sporesc eficiența procesului de predare/învățare, îmbunătățind atât performanța individuală cât și pe cea în echipă. Învățând să utilizeze eficient noile tehnologii, profesorii participanți la acest program au ocazia de a deprinde noi abilități de predare, pe care apoi le aplică în clasă, făcând astfel procesul de predare/învățare mult mai atrăgător pentru elevi.

Succesul Intel®Teach va fi evident în momentul în care profesorii vor dezvolta independent proiecte educaționale îmbunătățite cu ajutorul tehnologiei, concentrându-se pe abilitățile de învățare, nevoile și așteptările elevilor.

Radu Jugureanu, Florin Ilia

Demonstrarea CORECTITUDINII variantei RECURSIVE a ALGORITMILOR care folosesc metoda BACKTRACKING

Tiberiu Socaciu

Una dintre cele mai comode metode de rezolvare a unei probleme pentru care habar nu avem cum să găsim o soluție eficientă este backtracking-ul. Se știe că este un algoritm exponențial și este descurajant, dar argumentul pentru folosirea lui ar fi posibilitatea rulării acestuia pe un cluster de calculatoare[1], posibilitatea rafinării funcțiilor de continuare prin serializare spre a ne duce cât mai aproape de cea mai bună funcție de continuare[2]. Singura problemă care se mai pune în momentul în care ne lăsăm viața în mâinile acestui algoritm este: algoritmul backtracking este corect? Ne rezolvă el problema? Aici vom încerca să demonstrăm acest lucru pentru varianta recursivă.

Subprogramul[3] recursiv pentru algoritmul *Backtracking* este următorul:

```
algoritm btr (level, np, p, x)
  dacă level = np + 1 atunci
    apelează prelucrează_soluție (np, x)
  altfel
    pentru i ← 1, p_level execută
      x_level ← i
      dacă fi (level, x) atunci
        apelează btr (level + 1, np, p, x)
      sfârșit dacă
    sfârșit pentru
  sfârșit dacă
sfârșit algoritm
```

Apelul subprogramului se realizează prin:

```
apelează btr (1, np, p, x)
```

Vom demonstra mai întâi că orice rezultat furnizat de algoritmul `btr` este corect; cu alte cuvinte, fucția φ are valoarea *adevărat*.

[1] *este de remarcat faptul că recursivitatea poate fi un mijloc descurajant de paralelizare, vezi [5], [16]*

[2] *vezi [4], [6]*

[3] *subprogramul este descris în pseudocod, parametrii se transmit prin valoare*

Trebuie remarcat faptul că unicul loc în care are loc o ieșire a datelor este apelul `prelucrează_soluție`; dar acest apel nu poate avea loc decât dacă `level = np + 1`, adică suntem la cea de-a (np + 1) - a instanță a algoritmului `btr` (cel de-al `np` - lea autoapel), adică avem deja completate `np` valori ale vectorului `x`, valori care au trecut toate de testul de acceptanță prin apelul funcției de continuare `fi`; aceasta înseamnă că datele furnizate la ieșire de către algoritm reprezintă soluții.

Vom demonstra acum că nu există soluții pe care algoritmul `btr` nu le găsește.

Presupunem prin absurd că vectorul $y_1, y_2, \ldots, y_{np}$ reprezintă o soluție[4] corectă care nu a fost găsită de algoritmul `btr`.

Notăm prin `k` maximul numărului de elemente consecutive din vectorul `y`, începând cu prima poziție, care sunt identice cu elementele vectorului soluțiilor parțiale `x` de-a lungul perioadei de viață.

Evident, vom avea valoarea ***adevărat*** pentru `fi(k,` $y_1, y_2, \ldots, y_k, y_{k+1}$); așadar, după atribuirea $x_{k+1} \leftarrow y_{k+1}$, în urma apelului de evaluare a funcției `fi` se va efectua apelul recursiv intrându-se într-o nouă instanță a subprogramului `BTR`; astfel, avem `k + 1` valori identice, deci numărul `k` ales nu este valoarea maximă care are proprietatea cerută; am ajuns la o contradicție, deci ipoteza (`y` reprezintă o soluție corectă care nu a fost găsită de algoritmul `btr`) este falsă. Prin urmare, orice rezultat furnizat de algoritmul `btr` este corect.

Am demonstrat că algoritmul este parțial corect[5]; urmează să demonstrăm că este și finit[6].

Fie `pmax` maximul elementelor vectorului `p`. Definim funcția

$$W: \{1, 2, \ldots, np + 1\} \times M \rightarrow N,$$

unde `W(level, x)` este numărul format din `np` cifre, scris în baza `pmax`, cu primele `level` cifre din vectorul `x` și ultimele `np - level` cifre 0.

În mod evident `W` este o funcție limitată superior de valoarea $pmax^{np+1}$.[7] Această valoare este atinsă în cazul în care avem `level = np + 1`.

Rămâne de arătat faptul că funcția `W` este o funcție monoton crescătoare.

Acest lucru, însă, poate fi demonstrat imediat folosindu-se observația că, în cadrul algoritmului `btr`, înainte de apelul recursiv, are loc o creștere a valorii x_{level} și, implicit, a valorii funcției `W`, iar la apelul recursiv are loc o creștere a valorii `level`, deci, implicit, avem mai multe cifre în `W` și valoarea crește și în acest caz.

În concluzie, am demonstrat că algoritmul `btr` este total corect[8].

Bibliografie

1. **Tiberiu Socaciu,** *Învățați Backtracking. Exemple în Pascal și C*, Editura InfoData Cluj-Napoca, 2006, ISBN 973-87774-1-0, 160 de pagini;
2. **Tiberiu Socaciu,** *Metoda Backtracking cu exemple în Pascal*, Editura Edusoft Bacău, 2005, ISBN 973-87496-5-4, 100 de pagini;
3. **Tiberiu Socaciu,** *Metoda Backtracking cu exemple în C*, Editura Edusoft Bacău, 2005, ISBN 973-87496-6-2, 100 de pagini;
4. **Bogdan Pătruț, Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, comunicare la Linux și mediile virtuale de instruire, Arad, 11-15 septembrie 2006, eveniment organizat de Universitatea de Vest "Vasile Goldiș" Arad;
5. **Darius Bufnea, Tiberiu Socaciu,** *De la backtracking secvențial la cel concurent și paralel*, comunicare susținută la Zilele Unirii, Focșani, 22-24 ianuarie 2005;
6. **Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, GInfo nr. 16/8 (decembrie 2006), Editura Agora Media Târgu-Mureș;
7. **Tiberiu Socaciu,** *Cea mai bună funcție de continuare*, GInfo nr. 16/7 (noiembrie 2006), Editura Agora Media Târgu-Mureș;
8. **Tiberiu Socaciu,** *Serializarea ca metoda de determinare a funcției de continuare*, GInfo nr. 16/6 (octombrie 2006), Editura Agora Media Târgu-Mureș;
9. **Tiberiu Socaciu,** *O problemă de combinatorică*, GInfo nr. 11/4 (aprilie 2001), Editura Agora Media Târgu-Mureș;
10. **Tiberiu Socaciu,** *Metoda Backtracking*, if nr. 5/1992 (16), Editura Computer Press Agora Târgu-Mureș;
11. **Bogdan Pătruț, Tiberiu Socaciu,** *Algoritm Backtracking pentru un cluster bazat pe mesaje*, Studia Universitatis "Vasile Goldiș" Arad, seria Științe Exacte, nr. 16 (2006), în curs de apariție.
12. **Tiberiu Socaciu,** *Principiul contorului*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
13. **Tiberiu Socaciu,** *Rezolvarea unei probleme de backtracking*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
14. **Tiberiu Socaciu,** *Corectitudinea programelor*, teză de dizertație, Universitatea Babeș-Bolyai Cluj, Facultatea de Matematică și Informatică, 1996 (conducător științific: prof. univ. dr. Militon Frențiu);
15. **Tiberiu Socaciu,** *Demonstrarea corectitudinii algoritmilor care folosesc metoda backtracking*, GInfo nr. 17/1 (ianuarie 2007), Editura Agora Media Târgu-Mureș;
16. **Tiberiu Socaciu,** *Paralelism și concurență în algoritmul backtracking recursiv*, GInfo nr. 17/4 (aprilie 2007), Editura Agora Media Târgu-Mureș.

[4] *se verifică $\varphi(Y)$*
[5] *produce toate rezultatele în ipoteza că se termină*
[6] *se termină într-un număr finit de pași*
[7] *explicația se bazează pe transcrierea valorii numărului dat de W*
[8] *parțial corect și finit*

Algoritmi de NUMĂRARE

Ion Petre

În cadrul acestui articol vom prezenta mai multe probleme pentru a căror rezolvare este necesară cunoașterea anumitor algoritmi de numărare.

Naveta interplanetară

O navă spațială efectuează o cursă interplanetară plecând cu viteza inițială de 1 km/s. După fiecare secundă viteza navei se dublează instantaneu, păstrându-se constantă pe parcursul unei secunde. Știind că distanța dintre două planete este *d* km, să se determine în cât timp se parcurge această distanță.

Intervalul de timp trebuie exprimat în secunde (o secundă "începută" va fi numărată ca fiind o secundă întreagă).

Notă

Vom considera că se aplică legile mecanicii clasice. Nu suntem interesați de aspectele legate de imposibilitatea depășirii vitezei luminii sau de modul în care se compun vitezele atunci când acestea sunt apropiate de cea a luminii.

Listing NAVA.CPP

```
#include<fstream.h>

unsigned long d, t;

void main(void)
{
  ifstream f("nava.in");
  f >> d;
  f.close();

  t = log(d) / log(2);
  if ((1 << t) < d)
  {
    t++;
  }

  ofstream g("Nava.Out");
  g<<t;
  g.close();
}
```

Datele de intrare

Din fișierul **nava.in** se citește valoarea *d*.

Datele de ieșire

În fișierul **nava.out** se va scrie timpul necesar parcurgerii distanței *d*.

Restricție

- $1 < d \leq 2000000000$

Soluție

Soluția este foarte simplă, bazându-se pe o simplă formulă matematică. Implementarea este prezentată în casetă.

Numere și suma cifrelor

Se consideră un număr *n* care poate avea una dintre valorile 2, 4, 6 sau 8.

Câte numere formate din *n* cifre au suma primelor *n* / 2 cifre egală cu suma ultimelor *n* / 2 cifre? Se acceptă și numărul `0...0` (*n* cifre)

Soluție

Rezolvarea este destul de evidentă. Codul sursă este prezentat în casetă.

Produs maxim

Se consideră un număr natural nenul *Nr*. Să se descompună acest număr în sumă de numere naturale nenule astfel încât produsul acestora să fie maxim posibil.

Exemple

Intrare

```
Nr = 8
```

Ieșire

```
3 3 2
```

Listing SUMA.CPP (I)

```
#include<fstream.h>

int c[50];
int n;
int d;
int i, j, k, l;
long nr;

void main(void)
{
  do
  {
    cout << endl << "n= ";
    cin >> n;
  }
  while ((n < 2) || (n > 8) || (n % 2 == 1));

  memset(c, 0, sizeof(c));
  d = n / 2;

  switch (d)
  {
    case 1:
    {
      nr = 10;
      break;
    }

    case 2:
    {
      for (i = 0; i <= 9; i++)
      {
        for (j = 0; j <= 9; j++)
        {
          c[i + j]++;
        }
      }
      nr = 0;
      for (i = 0; i <= 18; i++)
      {
        nr = nr + c[i] * c[i];
      }
      break;
    }
```

Intrare
Nr = 9

Ieșire
3 3 3

Intrare
Nr = 10

Ieșire
3 3 2 2

Listing SUMA.CPP (II)

```
    case 3:
    {
      for (i = 0; i <= 9; i++)
      {
        for (j = 0; j <= 9; j++)
        {
          for (k = 0; k <= 9; k++)
            c[i + j + k]++;
        }
      }
      nr = 0;
      for (i = 0; i <= 27; i++)
      {
        nr = nr + c[i] * c[i];
      }
      break;
    }

    case 4:
    {
      for (i = 0; i <= 9; i++)
      {
        for (j = 0; j <= 9; j++)
        {
          for (k = 0; k <= 9; k++)
          {
            for (l = 0; l <= 9; l++)
            {
              c[i + j + k + l]++;
            }
          }
        }
      }
      nr = 0;
      for (i = 0; i <= 36; i++)
      {
        nr = nr + c[i] * c[i];
      }
      break;
    }
  }
  cout << endl << "Nr = " << nr;
}
```

Intrare
Nr = 11

Ieșire
3 3 3 2

Soluție

Se determină mai întâi de câte ori apare numărul 3 în descompunerea numărului dat: `o3 = Nr / 3`.

În continuare, se determină restul împărțirii la 3 al numărului dat: `r = Nr % 3`.

Dacă restul este 1, atunci se combină cu un 3 și ultimii doi termeni au valoarea 2.

Numărul aparițiilor unei cifre (I)

Să se determine, în mod eficient, de câte ori apare o cifră nenulă *cf* în scrierea tuturor numerelor naturale nenule mai mici sau egale cu un număr natural *nr*.

Se dau *n* perechi de forma (*nr*, *cf*) care trebuie citite din fișierul **cifre1.in** și, pentru fiecare pereche, rezultatul trebuie scris în fișierul **cifre1.out**.

Indicații

Observăm că:

- în intervalul [1, 9] fiecare cifră nenulă apare odată;
- în intervalul [1, 99] fiecare cifră nenulă apare de 20 de ori;
- în intervalul [1, 999] fiecare cifră nenulă apare de 300 de ori;
- în intervalul [1, 9999] fiecare cifră nenulă apare de 4000 de ori;
- ...

Se preia câte o cifră din număr (*ucf*), începând cu cea mai puțin semnificativă. Putem întâlni trei situații (fiecare dintre acestea trebuie tratată separat):

- *ucf* < *cf*;
- *ucf* > *cf*;
- *ucf* = *cf*.

Vom folosi următoarele notații:

- `exp`: va conține, în ordine, puterile lui 10: 10^0, 10^1, 10^2, 10^3, ...
- `coef`: va conține, în ordine, valorile 0, 1, 20, 300, 4000, ...
- `rnr`: numărul format din cifrele eliminate până la pasul curent;
- `nc`: numărul aparițiilor cifrei *cf*.

Exemplul #1

Pentru *nr* = 56 și *cf* = 5 obținem:

nc = (6 · 0 + 1) + // *ucf* = 6, *rnr* = 0, *coef* = 0, *exp* = 1
+ (5 · 1 + 6 + 1) // *ucf* = 5, *rnr* = 6, *coef* = 1, *exp* = 10

Observație

- *rnr* + 1 și *exp* indică numărul aparițiilor cifrei *cf* = 5 pe poziția curentă.

Exemplul #2

Pentru *nr* = 2756 și *cf* = 5 obținem:

nc = (6 · 0 + 1) + // *ucf* = 6, *rnr* = 0, *coef* = 0, *exp* = 1
+ (5 · 1 + 6 + 1) + // *ucf* = 5, *rnr* = 6, *coef* = 1, *exp* = 10
+ (7 · 20 + 100) + // *ucf* = 7, *rnr* = 56, *coef* = 20, *exp* = 100
+ (2 · 300) // *ucf* = 2, *rnr* = 756, *coef* = 300, *exp* = 1000

Formula de calcul a numărului aparițiilor este:

$$nc \leftarrow nc + ncf \cdot coef + \alpha.$$

Listing CIFRE1.CPP

```
#include<fstream.h>

unsigned nr;
unsigned rnr;
unsigned coef;
unsigned exp;
unsigned nc;
unsigned char cf;
unsigned char ucf;
int n;
int i;

void main(void)
{
  ifstream f("cifre1.in");
  ofstream g("cifre1.out");

  f >> n;

  for (i = 1; i <= n; i++)
  {
    f >> nr>>cf;
    nc = rnr = coef = 0;
    exp = 1;
    while(nr)
    {
      ucf = nr % 10;
      nr /= 10;
      if (ucf < cf)
      {
        nc = nc + ucf * coef;
      }
      else
      {
        if (ucf == cf)
        {
          nc = nc + ucf * coef + rnr + 1;
        }
        else
        {
          nc = nc + ucf * coef + exp;
        }
      }
      rnr = ucf * exp + rnr;
      coef = coef * 10 + exp;
      exp*=10;
    }
    g << nc << endl;
  }

  f.close();
  g.close();
}
```

Valoarea α este:

- 0 dacă *ucf* < *cf*;
- *exp* dacă *ucf* > *cf*;
- *rnr* + 1 dacă *ucf* = *cf*.

Implementarea este prezentată în casetă.

Listing NRCIF3.CPP

```
#include <iostream.h>
#include <math.h>

unsigned long Nr;
int nc;

void main (void)
{
  cout << endl << "Nr= ";
  cin >> Nr;
  nc = log10(Nr) + 1;
  cout << "Numarul cifrelor este "
       << nc << ".";
}
```

Numărul aparițiilor unei cifre (II)

Să se determine, în mod eficient, de câte ori apare o cifră nenulă *cf* în scrierea tuturor numerelor naturale din intervalul închis [*a* , *b*] (*a*, *b* ∈ N).

Se dau *n* triplete de forma (*cf*, *a*, *b*) care trebuie citite din fișierul **cifre2.in** și, pentru fiecare triplet, rezultatul trebuie scris în fișierul **cifre2.out**.

Soluție

Pentru fiecare triplet (*cf*, *a*, *b*) se determină numărul aparițiilor cifrei *cf* în intervalele [1, *a* - 1] și [1, *b*], apoi se calculează diferența rezultatelor. (Vezi problema de la secțiunea anterioară.)

Numărul cifrelor (I)

Se consideră un număr natural nenul *Nr*. Să se determine, în mod eficient, câte cifre are numărul dat.

Soluție

Numărul nc al cifrelor unui număr *Nr* este dat de partea întreagă a logaritmului zecimal al valorii *Nr*, la care se adaugă 1:

$$nc = [\lg Nr] + 1$$

Implementarea este prezentată în casetă.

Numărul cifrelor (II)

Se consideră *n* numere naturale formate din cel mult nouă cifre. Să se determine, în mod eficient, pentru fiecare element *Nr* al șirului, numărul total al cifrelor din care sunt formate numerele cuprinse între 1 și *Nr*.

Cele *n* numere trebuie citite din fișierul **nrcifre2.in** și, pentru fiecare număr, rezultatul trebuie scris în fișierul **nrcifre2.out**.

Soluția #1

Așa cum am afirmat și pentru o problemă anterioară, se poate observa că:

- în intervalul [1, 9] fiecare cifră nenulă apare odată;
- în intervalul [1, 99] fiecare cifră nenulă apare de 20 de ori;
- în intervalul [1, 999] fiecare cifră nenulă apare de 300 de ori;
- în intervalul [1, 9999] fiecare cifră nenulă apare de 4000 de ori;

...

Vom folosi următoarele notații:

- exp: va conține, în ordine, puterile lui 10: 10^0, 10^1, 10^2, 10^3, ...
- coef: va conține, în ordine, valorile 0, 1, 20, 300, 4000, ...
- rnr: numărul format din cifrele eliminate până la pasul curent;
- nrc: numărul cifrelor;
- rest: numărul cifrelor pentru intervalul pentru care se calculează coef;
- p: indicele cifrei prelucrate: 1, 2, 3, ...

Exemplu

Pentru *nr* = 7568 avem situația prezentată în tabelul de pe această pagină.

Implementarea este prezentată în casetă.

Soluția #2

Observăm că:

- în intervalul [1, 9] avem 1 · 9 = 9 cifre;
- în intervalul [1, 99] avem 2 · 90 = 180 de cifre;
- în intervalul [1, 999] avem 3 · 900 = 2700 de cifre;
- în intervalul [1, 9999] avem 4 · 9000 = 3600 de cifre;

...

Se elimină câte o cifră din număr, începând cu cea mai puțin semnificativă.

Vom folosi următoarele notații:

- ex: va conține, în ordine, numerele de forma 9, 99, 999, 9999, ...

ucf	nrc = (ucf - 1) · coef + (rnr + 1) · p + rest	rest = rest + 9 · coef	exp	coef
8	(8 - 1) · 1 + (0 + 1) ·1 + 0	-	1	1
6	(6 - 1) · 20 + (8 + 1) · 2 + 9	9 = 0 + 9 · 1	10	20
5	(5 - 1) · 300 + (68 + 1) · 3 + 189	189 = 9 + 9 · 20	100	300
7	(7 - 1) · 4000 + (568 + 1) · 4 + 2889 = **29165**	2889 = 189 + 9 · 300	1000	4000

Listing NRCIF2_1.CPP

```
#include<fstream.h>

unsigned nr;
unsigned rnr;
unsigned coef;
unsigned exp;
unsigned nrc;
unsigned p;
unsigned char ucf;

int n;
int i;

void main(void)
{
  ifstream f("nrcifre2.in");
  ofstream g("nrcifre2.out");
  f >> n;

  for (i = 1; i <= n; i++)
  {

    f >> nr;

    rest = rnr = p = 0;
    coef = exp = 1;

    while (nr)
    {

      ucf = nr % 10;
      nr /= 10;
      p++;
      nrc = (ucf - 1) * coef +
                        (rnr + 1) * p + rest;
      rnr = ucf * exp + rnr;
      rest = rest + 9 * coef;
      exp * = 10;
      coef = coef * 10 + exp;

    }

    g << nrc << endl;

  }

  f.close();
  g.close();
}
```

Listing NRCIF2_2.CPP

```
#include<fstream.h>

unsigned long nr;
unsigned long cnr;
unsigned long coef;
unsigned long ex;
unsigned long nrc;
unsigned long p;

int n;
int i;

void main(void)
{
  ifstream f("nrcifre2.in");
  ofstream g("nrcifre2.out");
  f >> n;

  for (i = 1; i <= n; i++)
  {

    f >> nr;

    cnr = nr;
    ex = nrc = 0;
    p = 1;
    coef = 9;

    while (cnr > 9)
    {

      nrc = nrc + p * coef;
      cnr /= 10;
      p++;
      coef = coef * 10;
      ex = ex * 10 + 9;

    }

    nrc += (p * (nr - ex));

    g << nrc << endl;

  }

  f.close();
  g.close();
}
```

p	coef	ex	nrc = nrc + p · coef	nrc = nrc + p · (nr - ex)
1	9	9	0 + 1 · 9 = 9	
2	90	99	9 + 2 · 90 = 198	
3	900	999	189 + 3 · 900 = 2889	
4	9000	9999	2889 + 4 · (7568 - 999) = 29165	

- coef: va conține, în ordine, valorile 0, 1, 20, 300, 4000, ...
- rnr: numărul format din cifrele eliminate până la pasul curent;
- nrc: numărul cifrelor;
- p: indicele cifrei prelucrate: 1, 2, 3, ...

Listing NRCIF1.CPP

```
#include<fstream.h>

unsigned np;
unsigned nc;
unsigned dif;
unsigned rnr;
unsigned exp;
unsigned nrc;
unsigned p;
int n;
int i;

void main(void)
{
  ifstream f("nrcifre3.in");
  ofstream g("nrcifre3.out");
  f >> n;

  for (i = 1; i <= n; i++)
  {
    f >> nrc;
    nc = np = 0;
    p = exp = 1;
    while (nc + p * 9 * exp <= nrc)
    {
      nc = nc + p * 9 * exp;
      np = np + 9 * exp;
      p++;
      exp*=10;
    }
    dif = nrc - nc;
    if (dif % p == 0)
    {
      np = np + dif / p;
    }
    else
    {
      np = 0;
    }
    g << np << "  ";
  }

  f.close();
  g.close();
}
```

Exemplu

Pentru *nr* = 7568 avem situația prezentată în tabelul de pe această pagină.

Implementarea este prezentată în casetă.

Numărul cifrelor (III)

Pentru scrierea paginilor unei cărți (1, 2, 3, ...) s-au folosit *nrc* cifre. Să se determine, în mod eficient, câte pagini are cartea.

Din fișierul text **nrcifre3.in** se citesc *n* valori pentru *nrc*. Pentru fiecare valoare a lui *nrc*, rezultatul trebuie scris în fișierul **nrcifre3.out**.

Dacă nu se poate determina exact numărul de pagini, atunci în fișierul de ieșire se va scrie valoarea 0.

Soluție

Vom utiliza următoarele notații:

- np: numărul de pagini căutat;
- nc: numărul de cifre curent;
- p: numărul cifrelor unui număr din tranșa curentă;
- dif: diferența dintre numărul de cifre dat (*nrc*) și numărul de cifre curent calculat (nc).

Se determină numărul de cifre (*nc*) din numerele cuprinse între 1 și *np*, cu $nc \le nrc$.

Dacă *dif* nu se împarte exact la *p* atunci nu se poate calcula exact numărul de pagini (*np* = 0).

Implementarea este prezentată în casetă.

Numărul cifrelor (IV)

(problemă propusă)

Se consideră *n* intervale de numere naturale [*a* , *b*]. Să se determine, în mod eficient, câte cifre sunt în toate numerele naturale din fiecare interval [*a* , *b*].

Din fișierul **nrcifre4.in** se citesc *n* perechi de numere naturale [*a*, *b*]. Rezultatele trebuie scrise în fișierul **nrcifre4.out**.

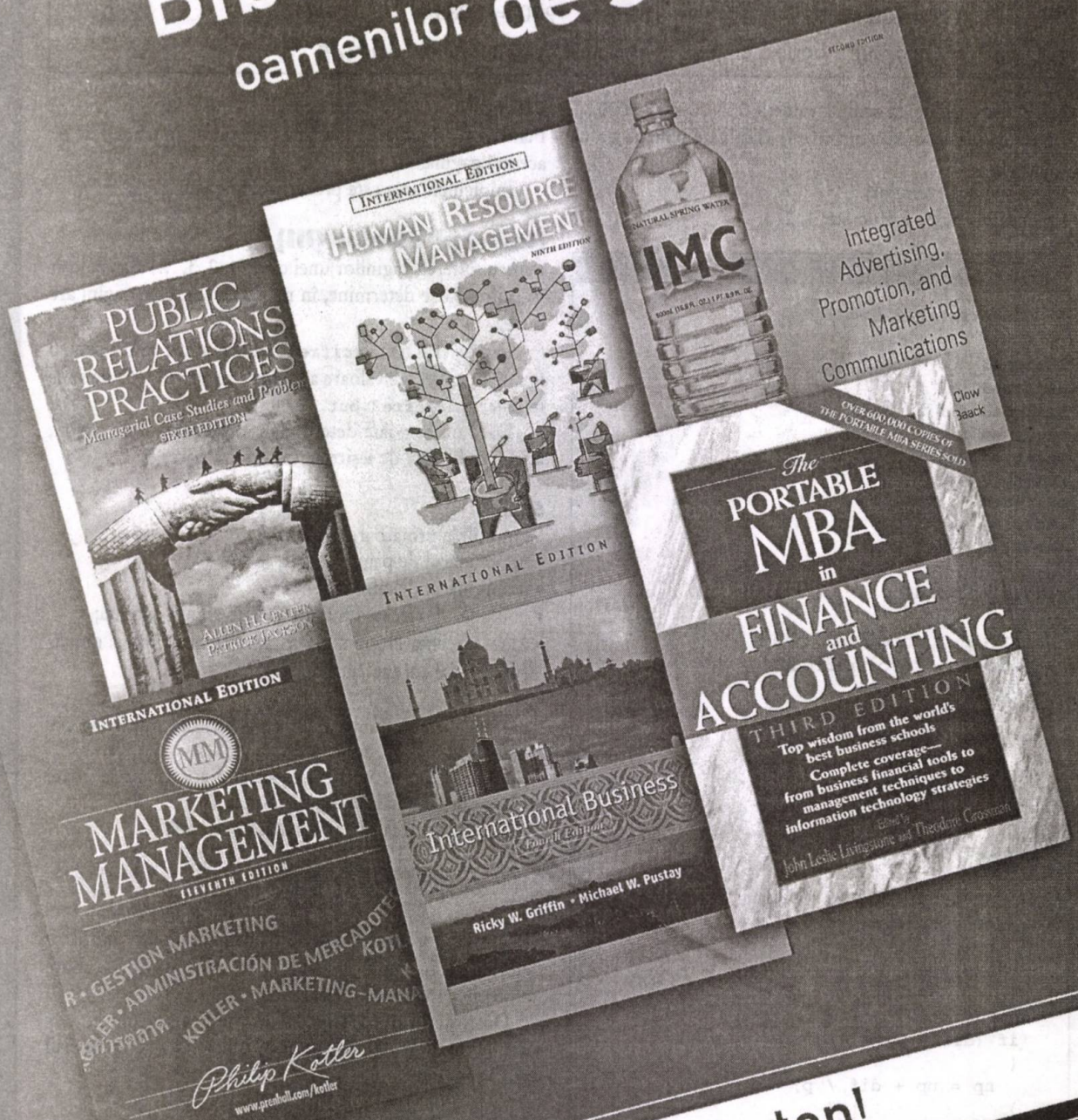
Numărul cifrelor (V)

(problemă propusă)

Se consideră *n* perechi de numere naturale *nc* și *a*. Pentru fiecare pereche de numere să se determine, în mod eficient, numărul natural *b* astfel încât numărul total de cifre ale numerelor naturale din intervalul închis [*a* , *b*] să fie *nc*.

Din fișierul **nrcifre5.in** se citesc *n* perechi de numere naturale *nc* și *a*. Rezultatele trebuie scrise în fișierul **nrcifre5.out**.

Biblioteca oamenilor de succes



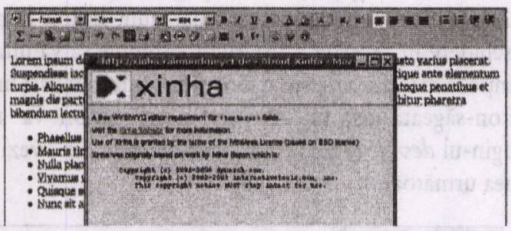
Învață să ajungi în top!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85 e-mail: dpastrama@byblos.ro
tel: 021-312.72.65, 0721-927.499
www.byblos.ro
BYBLOS

ÎNCORPORAREA unui editor WYSIWYG într-o pagină WEB

Vlad Bazon

În septembrie 1998 Microsoft lansa Internet Explorer 5.0-beta, care introducea elementul HTMLAREA; acesta, prin asociere cu un anumit script DHTML, permitea, în plus față de TEXTAREA, formatarea elementară a textului (ulterior, s-a renunțat la HTMLAREA). Apoi, în 2002 a apărut biblioteca HTMLAREA (creată de Mihai Bazon; ©interactivetools.com, ©dynarch.com; dezvoltată ulterior de Xinha), care permitea echiparea unui element textarea cu proprietăți și metode specifice unui editor WYSIWIG complet. Începând din 2005 s-au dezvoltat numeroase asemenea editoare WYSIWIG; dintre cele necomerciale (open-source) s-au impus tinyMCE, FCKeditor și Xinha (totdeauna, pentru adrese sau referiri puteți tasta în bara de adrese a browser-ului de exemplu, "google xinha", sau "google perl", etc.).

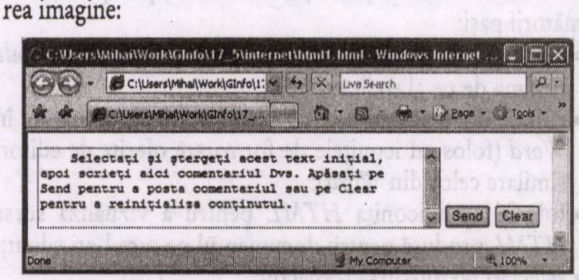


Elementul TEXTAREA

Elementul *HTML* **<TEXTAREA>** furnizează, în fond, un editor de text obișnuit:

```
<FORM
  action = "http://mySite.com/articole/
                                comentarii/read"
  method = "post"
>
  <TEXTAREA
    name = "textComment"
    rows = "5"
    cols = "50"
  >
    Selectați și ștergeți acest text inițial, apoi scrieți aici comentariul Dvs. Apăsați pe Send pentru a posta comentariul sau pe Clear pentru a reinițializa conținutul.
  </TEXTAREA>
  <INPUT type = "submit" value = "Send"/>
  <INPUT type = "reset" value = "Clear"/>
</FORM>
```

Acest document *HTML* va duce la afișarea în *browser* a unei pagini *Web* asemănătoare cu cea prezentată în următoarea imagine:



Programul indicat de **action** va prelua textul trimis (după efectuarea unui *click* pe butonul *Send*), îl va "curăța", eventual, de unele *tag*-uri și va scrie comentariul respectiv (într-o bază de date sau într-un fișier aflat pe server).

Biblioteca TinyMCE

Puteți descărca biblioteca *TinyMCE* de la **http://tinymce.moxiecode.com/**.

După dezarhivare obțineți un director `tinymce`; subdirectorul `examples` conține câteva fișiere `.htm` care arată diverse modalități de utilizare (încărcați într-un browser fișierul `example-simple.htm`; vizualizați textul-sursă).

Biblioteca propriu-zisă (care poate fi folosită pentru a crea în aplicațiile *Web* obiecte cu proprietățile și metodele necesare comportării asemănătoare unui editor *WYSIWIG*) este furnizată în subdirectorul `tiny_mce`; fișierele esențiale sunt `tiny_mce.js`, `blank.htm` precum și cele din subdirectorul `themes` (grupate în `themes/simple` și `themes/advanced`); pe lângă acestea, subdirectorul `plugins` cuprinde diverse module care extind posibilitățile editorului (adăugând metodele *JavaScript* și fișierele-șablon *CSS/HTML* necesare extinderii vizate), iar subdirectorul `langs` înregistrează *plugin*-urile care vor permite ca mesajele furnizate pe parcursul execuției să fie afișate într-o limbă sau alta (nu neapărat, în engleză).

Această structurare a bibliotecii (inițiată din 2002) a devenit deja standard.

Utilizări locale

TinyMCE poate fi exploatată și local (pe calculatorul propriu, fără a angaja resurse externe) - de exemplu pentru a crea fișiere *HTML* fără a folosi pe parcursul editării elementele proprii limbajului *HTML* (fără a cunoaște *HTML*).

De asemenea, biblioteca poate fi utilizată pentru a obține o variantă *HTML* a unui document care a fost realizat anterior folosind *Microsoft Word* (presupunând că nu suntem mulțumiți de modul în care *Word*-ul convertește documentele din formatul propriu în format *HTML*).

Pentru o folosire locală, nu mai este necesară nicio altă documentare: încărcați în *browser* fișierul `example_word.htm` (prezent în `/examples`); desigur, mai elegant ar fi ca, în prealabil, să încărcați fișierul respectiv într-un editor de text obișnuit (de exemplu, *Notepad*) și să ștergeți textele explicative conținute, precum și acele elemente care devin inutile dacă se lucrează local (cum ar fi `<INPUT type = "submit" name = "save" value = "Submit"/>`).

Pentru a testa capacitățile bibliotecii, puteți urma următorii pași:

- efectuați un *click* pe iconița *Toggle fullscreen mode* (ultima de pe al treilea rând);
- editați documentul așa cum obișnuiți când lucrați în *Word* (folosind iconițele de formatare oferite de editor, similare celor din *Word*);
- folosiți apoi iconița *HTML* pentru a vizualiza sursa *HTML* produsă pentru documentul pe care l-ați editat;
- selectați textul-sursă respectiv;
- copiați textul sursă folosind *CTRL + C*;
- deschideți un fișier-text;
- folosiți *CTRL + V* (*Paste*);
- apoi salvați acest ultim fișier cu extensia `.html`;
- deschideți fișierul în *browser*, pentru verificare.

Astfel, am obținut un document *HTML* valid, fără a ne folosi cunoștințele despre limbajul *HTML*.

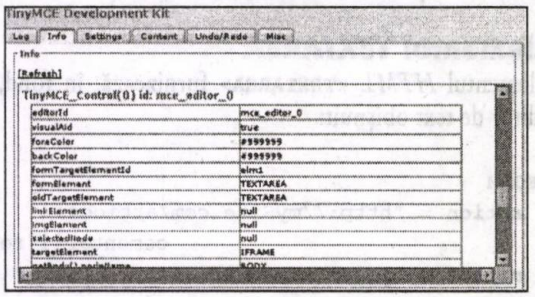
Pentru a obține pagina *HTML* care corespunde unui document existent `myDoc.doc`, creat folosind editorul *Microsoft Word* trebuie urmați următorii pași:

- deschideți în *Word* documentul inițial `myDoc.doc`;
- selectați conținutul documentului;
- copiați conținutul documentului (*Copy*);
- deschideți în *browser* fișierul `example_word.htm`,
- efectuați un click pe iconița *Paste from Word* (a cincea de pe al doilea rând);
- folosiți *CTRL + V* (*Paste*) în fereastra *pop-up* care a apărut;
- efectuați un *click* pe butonul *Insert* de la baza acestei ferestre.

Rămâne să folosiți, la fel ca în cazul exemplului precedent, iconița *HTML* pentru a obține fișierul `myDoc.html` corespunzător documentului `myDoc.doc` inițial.

Operațiile descrise pot fi utile în special persoanelor care nu au cunoștințe de *HTML*, dar au nevoie să producă un document *HTML* (deși *Word*-ul permite salvarea în format *HTML*, aceasta nu înseamnă că procedee de conversie precum cele de mai sus devin inutile: de exemplu, există persoane care nu dețin *Microsoft Office*, dar operează și cu fișiere `.DOC`).

Fișierul `example-full.htm` oferă și un instrument practic de investigație (inclusiv, pentru a vedea ce face *TinyMCE*); acest instrument este oarecum ascuns la prima deschidere în *browser* a fișierului - trebuie să vă uitați cu atenție la marginea de sus a ferestrei, unde veți găsi un mic buton-săgeată-jos; un *click* pe acest buton va activa plugin-ul *devkit* (*TinyMCE Development Kit* - vezi imaginea următoare).



În funcție de ingeniozitatea utilizatorului, se pot găsi și alte utilizări mai puțin banale; astfel, se poate realiza ușor (prin *Copy & Paste*) un colaj din părți ale unor pagini *Web*.

De exemplu, am putea alege de pe internet o pagină *Web* care încorporează un calendar (arată luna curentă și

are asociate butoane pentru trecere la altă lună); putem obține un calendar pentru întregul an efectuând următorii pași:

- deschidem, într-un nou *tab* al *browser*-ului, fișierul `example_word.htm`;
- definim (folosind iconița pentru tabele) un tabel cu 12 linii și 3 coloane;
- folosim iconița ***Paste from Word*** deschizând astfel o fereastră *pop-up* în care vom putea înscrie pe rând calendarele (chiar dacă acestea nu sunt documente *Word*);
- trecem în *tab*-ul în care avem pagina *Web* care conține calendarul;
- selectăm calendarul lunii ianuarie;
- revenim în tab-ul `example_word`;
- folosim *CTRL + V* (*Paste*);
- butonul ***Insert*** (astfel, calendarul lunii va fi înscris în acel câmp din tabel în care am poziționat în prealabil, cursorul);
- avansăm calendarul la luna următoare;
- repetăm operații și obținând, în final, calendarul pe un an.

Asupra calendarului obținut astfel putem efectua apoi diverse operații de formatare, folosind iconițele editorului; în final, selectăm tot, efectuăm *Copy & Paste* într-un fișier-text pe care îl salvăm apoi cu extensia `.HTML`.

Aplicație Web: testări

De obicei, la testare elevii sunt așezați câte unul în bancă și sunt supravegheați să nu copieze. Să extindem termenul și să imaginăm o testare liberă, cooperativă (mai degrabă, un cenaclu): profesorul formulează un anumit subiect și este dispus să acorde asistență fiecăruia dintre cei care vor dori să exerseze pe testul respectiv.

Profesorul și elevii nu sunt neapărat "de informatică", dar știu toți să utilizeze *Word* și să folosească un *browser*.

Să presupunem că am creat un site al școlii și vrem să îl extindem cu o rubrică de teste. Mai întâi trebuie să concepem (și să adăugăm bazei de date existente pe server) tabelele și relațiile adecvate pentru a înregistra subiecte, răspunsuri, pe cei care formulează subiecte și pe cei care răspund la teste. Apoi, trebuie realizate programele necesare pe server, pentru efectuarea următoarelor operații

- citerea și scrierea pentru tabelele respective;
- extragerea de informații din tabele;
- transmiterea, primirea și prelucrarea datele respective (către și dinspre clienții aplicației).

Pentru a concretiza câteva elemente ale aplicației, trebuie, până la urmă, să părăsim nivelul generic de formulare și să facem o alegere de limbaje și modele; ne luăm permisiunea de a invoca mai departe propria experiență de lucru asupra unui site școlar real, **http://lar.sitsco.com**.

Aplicația respectivă funcționează pe un server *Apache* și folosește ca limbaj de bază *perl* (o altă posibilitate ar fi fost, desigur, *PHP*); pentru lucrul cu baze de date se folosește *mySQL*.

Aplicația a fost inițial construită folosind *Catalyst*; acest pachet *perl* asigură integrarea și gestionarea componentelor de tip *Model de date*, *View* și *Controller*, furnizând un schelet inițial pentru aplicație.

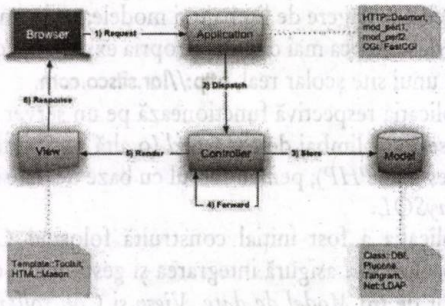
Apelul din linia de comandă `catalyst.pl MySite` construiește directorele specifice `MySite/lib` și `MySite/root`, rezultând un schelet abstract al aplicației, pe care apoi programatorul o va putea customiza după caz și o va putea extinde (în cazul de față, `MySite` este `lar`).

Catalyst asigură (scutindu-ne de o preocupare suplimentară) interacțiunea aplicației cu serverul și cu bazele de date, precum și repartizarea (*dispatch*) cererilor de resurse de la clienții aplicației către *controller*-ele de prelucrare și deservire corespunzătoare.

Listing testare.sql

```
CREATE TABLE teste (
  id integer unsigned NOT NULL auto_increment primary key,
    /* identifică unic o înregistrare */
  dataora varchar(16) default NULL,
    /* '20070424000000' pentru 2007-04-24 00:00:00 */
  id_user integer unsigned default NULL,
    /* referă în tabelul user, pe autorul testului */
  id_dom integer unsigned default NULL,
    /* has_a(id_dom => 'lar::Model::Domeniu' */
  tags varchar(128) default NULL,
    /* Titlu/Termeni-cheie */
  test text
    /* textul HTML al testului */
);

CREATE TABLE raspuns (
  id integer unsigned NOT NULL auto_increment primary key,
  dataora varchar(16) default NULL,
  id_user integer unsigned default NULL,
          /* has_a(id_user => 'lar::Model::User' */
  id_teste integer unsigned default NULL,
           /* has_a(id_teste => 'lar::Model::Teste' */
  resp text /* textul HTML al răspunsului */
);

CREATE TABLE domeniu (
  id integer unsigned NOT NULL auto_increment,
  nume varchar(16) default NULL,
       /* 'literatură', 'Arbori', 'Șah', 'Jocuri', etc. */
  PRIMARY KEY (`id`)
);
```

Imaginea următoare este preluată de la adresa http://www.catalystframework.org/.



Tabele și Modele (pentru date)

Mai întâi, creăm pe server un fișier-text `testare.sql`, în care definim (pentru *MySql*) tabelele necesare; sintaxa seamănă cu aceea din *FoxPro* (desigur, aici dăm un exemplu fictiv minimal; pot exista și alte câmpuri/relații). Înscrierea acestor tabele în baza de date existentă pe server se poate realiza cu o comandă de forma:

```
mysql -u utilizator -pparola baza-de-date
    < testare.sql
```

Fișierul `testare.sql` este prezentat în casetă.

Prin *Model* se vizează nu însăși baza de date, ci acele module de program care gestionează conexiunea cu baza de date și în fond, permit folosirea limbajului de bază (perl, în cazul de față) pentru operațiile specifice lucrului cu tabele.

De exemplu, pentru tabelul *Teste* creăm modulul `lib/lar/Model/Teste.pm` în care specificăm coloanele tabelului și relațiile (*has_a* și *has_many*) cu celelalte tabele. Codul sursă este prezentat în casetă.

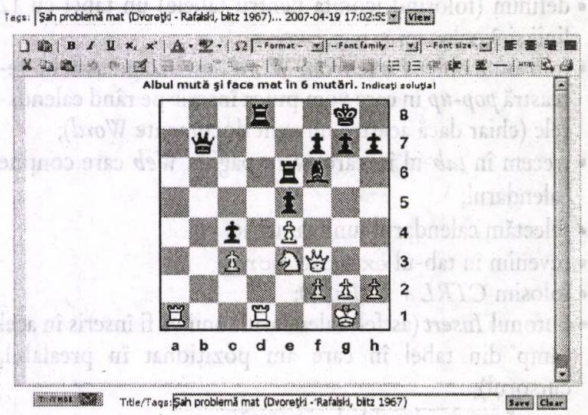
Un scenariu (posibil) de funcționare

Mai departe, înainte de a realiza subrutinele necesare (*controller*-ele), ar fi preferabil să imaginăm întâi un scenariu de interacțiune cu utilizatorul.

Altfel spus, realizăm un fișier `root/teste.html` care va conține elementele *HTML/ JavaScript* necesare acestei interacțiuni (un asemenea fișier ar ține de partea *View* a aplicației).

Catalyst permite folosirea a diverse sisteme de creare și procesare dinamică a conținutului unei pagini *Web*; am optat pentru *Template Toolkit*. Fișierul `root/teste.html` este prezentat în casetă.

În imaginea următoare este prezentat efectul obținut în urma unui *click* pe butonul *View* al formularului.



Utilizatorul alege unul dintre teste selectându-l din lista derulantă etichetată *Tags*; această listă corespunde elementului **<SELECT>** din formular.

În urma selecției efectuate, acest element reține în atributul **value** valoarea `teste.id` (scrisă aici ca pentru *MySql*). După ce a selectat testul, utilizatorul efectuează un *click* pe butonul *View*, declanșând astfel execuția codului *JavaScript* specificat în **onclick** la elementul **<INPUT>** al formularului din **teste.html**.

În esență, apelul `Ajax.Updater()` - prin care se invocă o funcție din biblioteca `prototype.js`, inclusă și aceasta în aplicație (și prezentată în cadrul articolului publicat în *GInfo 17 /4 - aprilie 2007*) - are următorul efect: se realizează o comunicare asincronă între *browser* și *server* prin care se cere execuția funcției `view()` pentru parametrul `sti` (= `teste.id`), funcție care este localizată în *controller*-ul `qteste`; după ce serverul returnează către *browser* rezultatul execuției, `Ajax.updater()` continuă prin a evalua scripturile existente în cadrul documentului primit (le execută) și, în final, scrie rezultatul în diviziunea indicată în `teste.html` prin **id** = "dest-test".

Se poate deduce opțiunea pe care am ales-o pentru sarcinile repartizate funcției `qteste/view()`: nu numai că va accesa înregistrarea corespunzătoare testului solicitat

Listing Teste.pm

```
package lar::Model::Teste;
use strict; use warnings;

use base 'lar::Model::Qdbi';
# Qdbi "include" Class::DBI ca model de bază și asigură conexiunea cu baza de date.

__PACKAGE__->table ('teste');
__PACKAGE__->columns(All => qw/ id dataora id_user id_dom tags test /);
__PACKAGE__->has_a(id_user => 'lar::Model::User');
__PACKAGE__->has_a(id_dom => 'lar::Model::Domeniu');
__PACKAGE__->has_many(raspuns => 'lar::Model::Raspuns' => 'id_teste');
```

Listing teste.html (I)

```
<SCRIPT>
  // inițializează un obiect-editor tinyMCE,
  // invocând metoda init()
  // mode: "none" editorul nu este asociat
  // încă, niciunui element HTML

  tinyMCE.init({
    width : "630",
    height: "400",
    mode : "none",
    theme : "advanced",
    theme_advanced_toolbar_location : "top",
    plugins : "paste, preview, table, print",
    theme_advanced_buttons1 :
      "newdocument,pasteword,separator,bold,
       italic,underline,sub,sup,separator,
       forecolor,backcolor,separator,
       charmap,formatselect,fontselect,
       fontsizeselect,separator,justifyleft,
       justifycenter,justifyright,
       justifyfull",
    theme_advanced_buttons2 :
      "cut,copy,paste,separator,undo,redo,
      separator,tablecontrols,separator,
      bullist,numlist,outdent,indent,
      separator,image,hr,separator,code,
      preview,print",
    theme_advanced_buttons3 : "",
    button_tile_map : true,
    cleanup_on_startup : true,
    cleanup: true,
    auto_reset_designmode : true
  });
</SCRIPT>
```

Listing teste.html (II)

```
<!-- obține înregistrările din "teste" și le
ordonează descrescător după teste.dataora -->
[% USE teste1 = Class('lar::Model::Teste') %]
[% teste2 = teste1.retrieve_all %]
[% teste = teste2.sort('dataora').reverse %]

<FORM id = "f-sti">
  <P>
  <SPAN class = "item">
    Tags:
  </SPAN>
  <SELECT name = "sstiri" id = "ttst">
    [% FOREACH sti = teste %]
      <OPTION value = "[%sti.id%]">
        [% sti.tags %] ... [% sti.dtf %]
      </OPTION>
    [% END %]
  </SELECT>
  <INPUT
    type = "button"
    class = "butsub"
    value = " View "
    onClick =
      "var sti=this.form.sstiri.value;
       new Ajax.Updater(
         'dest-test',
         '/qteste/view/' + sti,
         { asynchronous:true,
           parameters:'',
           evalScripts:true
         }
       );"
  />
  </P>
</FORM>

<DIV id = "dest-test">
</DIV>
```

(pentru a o transmite înapoi *browser*-ului drept răspuns), dar va încorpora în răspuns - pe lângă conținutul testului - și un *script* prin care (atunci când va fi executat în *browser*) să se instanțieze un editor *TinyMCE* (al cărui conținut să devină testul solicitat de *browser*).

Ar fi de menționat faptul că încercările de a soluționa mai obișnuit, problemele de integrare între *prototype-Ajax* și *tinyMCE* au eșuat; cauza pare a fi faptul că pe *site*-ul respectiv există două *tab*-uri, în care trebuie create editoare *tinyMCE* independente între ele.

Controller-ul Qteste

Pentru a asigura (sub *Catalyst*) funcționalitatea descrisă mai sus, trebuie să creăm modulul `Qteste.pm`, în care să definim în primul rând funcția `view()`. Implementarea este prezentată în casetă.

Click-ul pe butonul ***Save*** va fi interceptat (întâi) de *TinyMCE*, prin metoda sa `triggerSave()`; această funcție efectuează (având setat `cleanup: true`) unele "curățiri" ale conținutului (de exemplu, poate elimina sau transforma *tag*-uri precum **<SCRIPT>** pentru a evita prezența unor porțiuni de cod "periculos" în documentul salvat), apelează, dacă există, funcțiile înregistrate de către programator în `tinyMCE.init({})` prin `cleanup_callback` sau `save_callback` și transferă controlul către funcția indicată în atributul **action** al formularului - în cazul nostru, către funcția `save_tmce()` din *controller*-ul `qteste`.

Ca logică, `save_tmce()` trebuie să verifice dacă utilizatorul are dreptul de a salva pe server (aplicația trebuie să prevadă tabele și *controller*-e pentru înregistrare/verificare de conturi și de roluri) și în acest caz, mai trebuie să scrie și să actualizeze conținutul transmis în tabelul potrivit.

Listing Qteste.pm (I)

```
package lar::Controller::Qteste;
use strict;
use warnings;
use base 'Catalyst::Controller';

sub view : Local {
  my ($self, $c, $is) = @_;

# în Perl, @_ este un tablou care conține
# parametrii transmiși subrutinei
# Catalyst fixează ca prim element din @_ o
# referință la modulul "parent"
# (la fel cum în C++ funcția-membru are un
# pointer ascuns "this" la clasa "parent")
# $c ("context") este un obiect Catalyst care
# conține informații despre cererea curentă
# în Perl, my localizează variabilele în
# interiorul unui bloc de cod

  my $st = lar::Model::Teste->retrieve($is);
# $st este obiect Class::DBI pentru acces la
# înregistrarea din teste de id=$is

# $out înregistrează o definiție de formular
# pentru instanțierea unui editor tinyMCE
# definiția e cuprinsă între etichetele
# defFORM (pentru lizibilitate)
# '.' este operatorul de concatenare (perl)

my $out = "defFORM";
<FORM
  method = 'post'
  action = 'qteste/save_tmce'
>
  <TEXTAREA
    id = 'elm2'
    name = 'elm2'
    rows = '20'
    cols = '60'>
  </TEXTAREA>
  <INPUT
    type = 'hidden'
    id='totem'
    name='totem'
    value = '".$st->id."\'
  />
```

Listing Qteste.pm (II)

```
  Title/Tags:
  <INPUT
    type = 'text'
    id = 'ttlu'
    name = 'ttlu'
    value = '".$st->tags."\'
    size='60'
  />
  <INPUT
    type = 'submit'
    class = 'butsub'
    name = 'save'
    value = 'Save'
    onclick =
      "tinyMCE.execCommand('mceRemoveControl',
                                  false,'elm2');\"
  />
  <INPUT
    type = 'button'
    class='butsub'
    value='Clear'
    onclick=\"\$('totem').value='0';
                        \$('ttlu').value='';\"
  />
</FORM>";
defFORM
# $('totem') este "prescurtarea" din
# prototype.js pentru
# document.getElementById()

# se adaugă un script pentru
# activarea/dezactivarea editorului (specific
# pentru tinyMCE):

$out .=
  "<script>var tmc =
        tinyMCE.getInstanceById('elm2');";
$out .=
  "tinyMCE.execCommand('mceAddControl',
                              false,'elm2');";
$out .=
  "tinyMCE.setContent('.'\''.$st->test.'\');";
$out .= "</script>";

# în final, se returnează răspunsul $out către
# browser:
$c->response->body($out);
}
```

MANAGEMENTUL CLASEI INFORMATIZATE CU

NetOp® School

Multe instituții de învățământ pun din ce în ce mai mult accentul pe educația asistată de calculator. Scopul acestui articol este de a prezenta schematic funcțiile de bază ale unui Sistem de management al claselor în rețea și să ofere două soluții simple.

Instalarea unor clase standard

Într-o școală cu clase dotate cu rețele locale de calculatoare, un sistem de management al claselor reprezintă un instrument valoros pentru profesori și studenți. Profesorul poate controla toate calculatoarele din scaunul lui confortabil, iar monitoarele pot fi partajate de toți pentru a crea un proces de învățare interactiv. Modulul ***Profesor NetOp*** se instalează pe calculatorul profesorului. Modulul ***Student NetOp*** se instalează pe calculatoarele tuturor elevilor. Modulele care utilizează același "Nume de clasă" vor fi conectate.

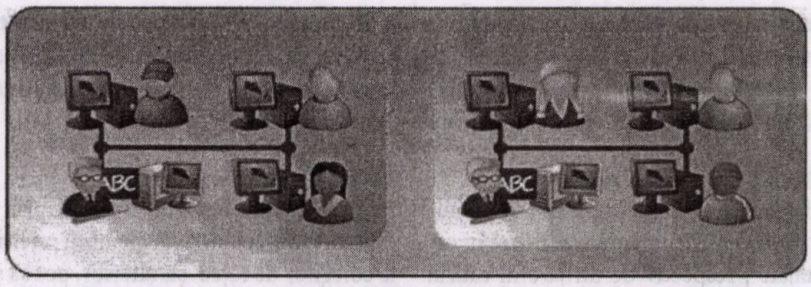


Fig. 1: Clase standard: un Profesor NetOp conectat la un număr de Studenți NetOp

Clasele standard au elevii și profesorul împreună. O clasă poate fi descrisă prin: nume, profesor, un număr de elevi.

Instalarea unor clase virtuale

O clasă poate fi virtuală în rețea (locală și/sau Internet), când unii sunt prezenți în aceeași cameră, iar alții sunt conectați la rețea din alte locații.

Caracteristici de bază ale produsului

Conectare. NetOp School permite profesorilor și elevilor să se conecteze în diferite moduri: a) Profesorii adaugă automat studenți la o anumită clasă; b) Elevii aleg o clasă dintr-o listă de clase active

Supraveghere. NetOp School asigură o supraveghere extensivă a elevilor. Dintr-o singură privire, profesorul poate urmări monitoarele elevilor și poate prelua controlul tastaturii și mous-lui pentru a oferi ajutor.

Partajare. NetOpSchool asigură tuturor elevilor un loc în primul rând, prin vizualizarea simultană a monitorului profesorului pe propriul monitor.

Reguli. În NetOp School se pot aplica reguli pentru a împiedica accesul elevilor la anumite programe de pe calculatorul propriu ori la anumite site-uri din Internet. Tastatura și mouse-ul pot fi blocate de profesor.

Teste. NetOp School are un modul de teste cu 10 tipuri diferite de întrebări, printre care: alegeri multiple, potrivirea textului cu imaginea, eseu, formule de încheiere, etc.

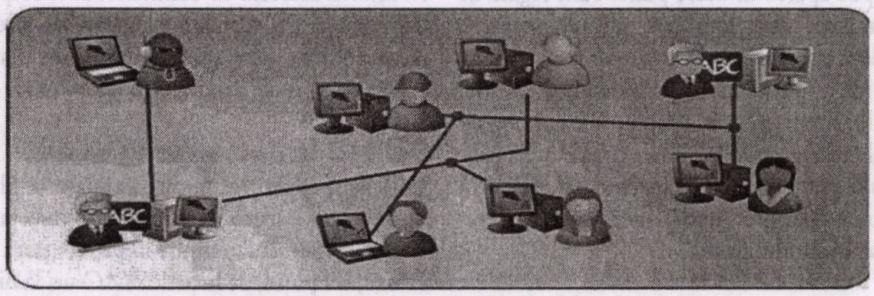


Fig. 2: Clasa virtuală: Profesorii NetOp oferă cursuri și Studenții NetOp selectează cursul dorit

Utilizatori români: American International School of Bucharest, Universitățile: Construcții București, Petru Maior Tg Mures, Transilvania Brașov, etc.

Informații suplimentare: www.netop.com, www.ps.ro

Reprezentant în România: Prosoft++ srl, sales@ps.ro

Tehnici de PROTECȚIE ANTI-FRAUDĂ în rețelele de cartier

Tiberiu Socaciu

Una dintre cele mai mari probleme în rețelele de cartier sau în marile rețele LAN este cea a protecției anti-fraudă: ce se întâmplă dacă cineva își asumă alt rol în cadrul unei rețele? Furtul de IP nedemonstrabil poate fi dezastruos din punctul de vedere al celui care administrează afacerea sau poate fi neplăcut din punctul de vedere al administratorului de rețea într-un LAN.

Asocierea MAC-IP cu ARP static

Am văzut că fiecărui adaptor de rețea i se atașează un număr unic[1] care joacă rolul de instrument de adresare în cadrul unei rețele partajate[2], respectiv de dirijare în cadrul unei rețele comutate[3].

Prin urmare, una dintre cele mai simple metode de protecție este cea bazată pe *ARP* static la nivelul ruterului de acces[4].

Asocierea MAC-IP la nivel de firewall

O metodă mai sigură de protecție împotriva furtului de identitate este asocierea *MAC-IP* la nivelul *firewall*-ului. Problemele care pot apărea în acest punct sunt legate de numărul mediu/maxim de reguli care trebuie parcurse pentru a decide dacă un pachet este valid sau nu.

O altă problemă ar putea apărea în momentul în care, din cauza traficului intens și a numărului mare de reguli prin care trec pachetele, avem o încărcare prea mare a procesorului. Sub *Linux* putem afla încărcarea procesorului prin afișarea conținutului fișierului `/etc/loadavg`[5]. Un exemplu[6] este prezentat în continuare:

```
[root@server carti]# cat /proc/loadavg
0.13 0.15 0.10 2/78 322
```

[1] *MAC-ul*
[2] *în rețelele cu BNC sau hub-uri*
[3] *în rețelele cu switch-uri*
[4] *numim ruter de acces, echipamentul pe care îl vedem în mod implicit ca poartă de ieșire în internet în momentul conectării la rețea (conectare permanentă sau temporară)*
[5] *primul număr reprezintă încărcarea momentană a procesorului*
[6] *exemplul este rulat pe un sistem Fedora Core 5 cu update-urile la zi*

Furtul de MAC

Evident, orice soluție de protecție prin asocierea *MAC-IP* este sortită eșecului dacă membrii rețelei pot afla *MAC*-urile colegilor de rețea[7] și, ulterior, pot să și le asume folosind operațiunea de *clonare*[8]. Evident, înainte de a efectua operația, vor trebui să verifice dacă "victimele" sunt sau nu active în rețea folosind aplicații de tipul `arping`.

Protecția cât mai aproape de client

Este imperios necesar a se folosi metode mai avansate de protecție. Prin urmare, este nevoie să se folosească metode de protecție cât mai aproape de client, pentru a nu da posibilitatea acestuia să comită fraude, respectiv pentru a-l depista cât mai repede posibil pe cel care comite frauda. Acest loc este echipamentul în care se conectează clientul rețelei[9], de regulă *switch*-ul sau modemul de cablu/*DSL*.

Switch-uri cu management

Cel mai apropiat loc în care se poate amplasa protecția este, de regulă, *switch*-ul în care este conectat clientul rețelei, dacă echipamentul permite implementarea de reguli *MAC-IP* sau, cel puțin, de liste de *MAC*-uri la nivel de port. Administrarea acestor switch-uri se realizează, de regulă, printr-o interfață *Web*[10] sau prin intermediul unei console seriale[11].

[7] *folosind interogări ale tabelei de MAC-uri*
[8] *presupunând că au drepturi administrative*
[9] *de regulă, o interfață Ethernet*
[10] *în acest caz, echipamentul în sine are, la rândul său, un IP accesibil via LAN*
[11] *configurate, de regulă, ca 8N1 pentru vitezele în intervalul 2400-19200*

Rezolvarea în cazul rețelelor CaTV

În momentul sincronizării unui modem de cablu *DOCSIS*, la primirea de către acesta a adresei *IP*[12], el primește și un fișier de configurare[13] în a cărui structură se regăsesc și următoarele câmpuri[14]:

- *Customer Premises Equipment (CPE) Ethernet MAC*;
- *Maximum Number of Hosts on the Ethernet port of the CM (CPEs)*; valorile posibile sunt cuprinse între 1 și 255; valoarea implicită este 1.

Prin urmare, există posibilitatea "legării" unor informații client de modemul de cablu pentru a se preîntampina furtul de informații. De asemenea, există versiuni de modemuri de cablu cu functionalități extinse datorită *firmware*-ului implementat și se pot adăuga anumite setări de protecție[15] prin metode specifice, cum ar fi modemuri din familia *uBR Cisco* 904/905/924/925, care au o versiune de *firmware* bazată pe *IOS* 12 sau modemuri accesibile prin interfața *Web* sau `telnet`.

Rezolvarea în cazul xDSL

De regulă, în cazul modemurilor *xDSL*, protecția la nivelul modemului se realizează prin interfața *Web* sau `telnet`, după ce acesta își ridică interfața corespunzătoare. Datorita faptului că legăturile de tip *xDSL* sunt de forma *punct-la-punct*[16] (DSLAM-modem xDSL), eventualele protecții se pot extinde la nivelul portului din *DSLAM* în care se leagă perechile de fire telefonice. Dar, cum interfața *WAN* la tehnologiile de tip *DSL* este perechea de fire[17], nu prea există mecanisme de protecție, fiind necesară o protecție la nivel superior (de regulă *PPPoE*).

Rezolvarea în cazul rețelelor Wireless

La nivelul *wireless* protecțiile bazate pe *MAC*[18]-ul partenerului nu sunt întotdeauna eficiente, deoarece există posibilitatea clonării *MAC*-ului pe foarte multe echipamente, respectiv există posibilitatea modificării *firmware*-ului cu variante de *Linux*[19], având practic libertate deplină în manipularea echipamentului[20]. În aceste cazuri, soluția constă în criptarea conexiunilor radio.

"Ruperea" rețelei de cartier

În general, soluțiile bazate pe echipamente cu management sunt mai scumpe și prohibitive pentru o rețea de cartier. Pentru acestea, o soluție ar putea consta în introducerea de rutere de acces la câțiva clienți și rutarea claselor de *IP*-uri către acele rutere. În acest caz, eventualele fraude se pot limita și se poate reduce nivelul lor de propagare. Pe post de rutere de acces se pot folosi cu succes sisteme *Pentium* cu minim două plăci de rețea sau echipamente gen *Cisco Linksys* cărora le poate fi rescris *firmware*-ul cu versiuni de *Linux* specifice[21]. Unul dintre avantajele folosirii acestor soluții ar fi crearea de inele logice prin legarea într-o topologie redundantă a minim două rutere de acces și instalarea unor protocoale specifice pentru propagarea rutelor[22].

Alternativa PPPoE

O altă soluție aplicată de furnizorii de internet este cea bazată pe crearea de legături *punct-la-punct* între un echipament al furnizorului și calculatoarele clientului din rețea, prin ridicarea unor legături de date de tipul *PPP*[23]. Legăturile *PPP* sunt uzual folosite în cazul legăturilor de tip *dial-up*, unde ruterul de acces oferă în momentul conectării un *IP* și o rută implicite, pe baza unor date de identificare[24] verificabile într-o baza de date[25]. Echipamentul care va fi server de acces trebuie să fie suficient de robust pentru a rezista la încărcarea procesorului cauzată de numărul uneori imens de rute și de interfețe care sunt ridicate[26]. Practica a arătat că un sistem *Pentium* 4 cu procesor de 3 GHz rezistă la aproximativ 1000 de clienți. Prin urmare, este recomandabil ca gestiunea rutelor să fie realizată folosind echipamente care să fie în concordanță cu numărul de clienți din rețea.

Bibliografie

1. **Tiberiu Socaciu**, *MAC și ARP (I)*, *GInfo* 17/2, februarie 2007, Editura Agora Media, Târgu Mureș;
2. **Tiberiu Socaciu**, *MAC și ARP (II)*, *GInfo* 17/3, martie 2007, Editura Agora Media, Târgu Mureș;
3. **Tiberiu Socaciu**, *VLAN, TRUNK și STP*, *GInfo* 17/1, ianuarie 2007, Editura Agora Media, Târgu Mureș;
4. **Tiberiu Socaciu**, *Tuneluri și bridge-uri*, *GInfo* 16/8, decembrie 2006, Editura Agora Media, Târgu Mureș;
5. **Tiberiu Socaciu**, *Limitări de viteză în rețele*, *GInfo* 16/6, octombrie 2006, Editura Agora Media, Târgu Mureș;
6. **Tiberiu Socaciu**, *Informatica privită... altfel sau 16 chestiuni de matematică, teorie, programare de sistem și networking*, Editura Edusoft, Bacău, 2006;
7. **Tiberiu Socaciu**, *Informatica privită... altfel sau câteva chestiuni de matematică, teorie, programare de sistem și networking*, Editura InfoData, Cluj-Napoca, 2006;
8. **Tiberiu Socaciu**, *O altfel de ... Informatică sau diverse chestiuni de matematică, backtracking, clustere de calculatoare, Linux, inteligență artificială și networking*, Editura InfoData, Cluj-Napoca, 2007.

[12] *via DHCP*
[13] *via TFTP*
[14] *vezi http://www.cisco.com/warp/public/109/dhcp_and_docsis_config_cm.html*
[15] *și nu numai; de exemplu, facilitate de rutare sau de translatare de adrese (SNAT/DNAT)*
[16] *așadar, există o certitudine minimală asupra capătului de rețea*
[17] *nu avem în mod obligatoriu o interfață bazată pe adrese MAC*
[18] *și, eventual, SSID-ului în cazul AP-Client sau Ad-hoc, dar nenecesar în cazul WDS*
[19] *cum ar fi OpenWRT; vezi http://openwrt.org/*
[20] *de exemplu, ruterele Cisco Linksys WRT 54G cu versiunile 1-4 de firmware sau Cisco Linksys WRT 54GL*
[21] *vezi secțiunea anterioară*
[22] *de exemplu, BGP*
[23] *Point to Point Protocol*
[24] *nume de utilizator și parolă*
[25] *aflată de cele mai multe ori ca fișier plain text, cu parolele eventual criptate sau ca tabelă mysql sau ca serviciu de autentificare RADIUS aflat pe echipament sau în rețea cu acesta*
[26] *Practic, pentru fiecare client/calculator al clientului i se alocă o interfață și se adaugă o rută în tabela de rutare dacă autentificarea se finalizează cu succes*

Programare gentică!

Generarea AUTOMATĂ a PROGRAMELOR de calculator

Oana Muntean

Programarea genetică este o metodă care, pornind de la cerințele problemei, creează automat un program pentru rezolvarea acesteia. De ce am avea nevoie de acest lucru? Răspunsul este unul cât se poate de simplu și îl poate da orice persoană care a încercat să scrie o aplicație: programarea nu este o sarcină ușoară. De aceea, găsirea unei modalități prin care calculatorul să fie capabil, de unul singur, să construiască un program pentru rezolvarea diferitelor probleme ar avea o importanță deosebită. Acest articol ilustrează conceptele cele mai semnificative din programarea genetică.

Introducere

"Cum ar putea calculatoarele să învețe să rezolve probleme fără a fi programate în mod explicit pentru acest lucru? În ce fel ar putea fi calculatoarele capabile să rezolve ceea ce este necesar fară a li se spune exact cum trebuie realizat acest lucru?"

Arthur Samuel (1959).

Una dintre cele mai mari și mai vechi provocări ale informaticii constă în conceperea unor mașini capabile de a manifesta un comportament *inteligent*. Avem nevoie de acest tip de comportament deoarece există nenumărate probleme pentru care nu avem metode exacte și rapide de rezolvare. *Jocul de șah* poate fi considerat a fi un exemplu în acest sens. Același lucru se poate spune despre *problema comis voiajorului*.

Scrierea de programe de calculator este un alt exemplu în care este necesară inteligență. Problemele de informatică sunt atât de variate încât nu există nicio modalitate de a le încadra într-un prototip general.

Au avut loc nenumărate încercări de automatizare a procesului de scriere a codurilor sursă. Printre ele se numără și ***programarea genetică*** (*PG*), care aparține unui set mai larg de tehnici cu inspirație din biologie și mai ales din teoriile lui *Darwin*.

Aceste teorii conțin trei principii pe baza cărora se susține că au evoluat speciile pe *Pământ*:

- trăsături ale părinților sunt moștenite de către descendenți;
- un proces denumit ***selecție naturală*** alege indivizii mai bine adaptați la mediu,
- variațiile, acumulate în timp, pot duce la formarea de noi specii.

Pe lângă programarea genetică mai există și alte tehnici care folosesc aceste principii: *algoritmii genetici*, *strategiile evolutive* și *programarea evolutivă*. Scopul principal al acestor trei tehnici a fost rezolvarea unor probleme de optimizare. De obicei se lucrează cu vectori de numere reale, întregi sau binare.

În schimb, în cazul programării genetice situația este complet diferită. Nu se mai dorește obținerea unor soluții simple la probleme de optimizare, ci se dorește obținerea unor programe de calculator capabile să rezolve o problema dată.

De exemplu, dacă scopul algoritmilor genetici este de a obține o soluție pentru o intrare particulară la problema comis voiajorului, atunci scopul programării genetice este de a obține un program de calculator capabil să rezolve toate instanțele acestei probleme.

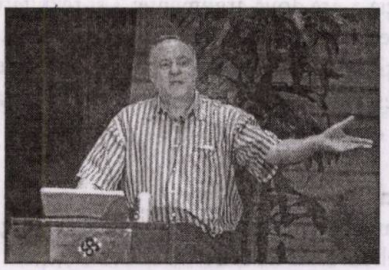


Figura 1: John Koza - cel mai prolific cercetător în domeniul programării genetice

Fondatorul acestei noi ramuri a *inteligenței artificiale* este cercetătorul american *John Koza* (vezi figura 1).

El a început cercetările pe această temă la începutul anilor '90.

Inițial, el a folosit limbajul *LISP* ca modalitate de reprezentare a soluțiilor programării genetice.

Ulterior au fost realizate implementări eficiente în toate limbajele de programare.

Cu timpul, s-a dovedit că ideea lui *Koza* funcționează pentru un număr foarte mare de probleme de inteligență artificială.

Astfel, tot mai mulți cercetători au devenit interesați de acest domeniu. În prezent există numeroase conferințe, cărți și reviste dedicate programării genetice.

Prezentul articol este organizat după cum urmează: în următoarea secțiune sunt prezentate principalele elemente ale programării genetice; urmează o secțiune în care sunt reproduși pașii unui algoritm de programare genetică; în continuare sunt dezvăluite câteva variante alternative de reprezentare a soluțiilor; urmează o secțiune în care se realizează o clasificare a celor mai importante aplicații unde a fost folosită programarea genetică; ultima secțiune prezintă avantajele și dezavantajele folosirii acestei metode.

Elementele principale ale unui algoritm de programare genetică

Din punctul de vedere al programării genetice rezolvarea problemelor nu presupune "construirea" unui program în sensul pe care informaticienii îl dau acestui lucru, ci se bazează doar pe o căutare a celui mai potrivit program din spațiul tuturor programelor posibile.

Această căutare se efectuează prin intermediul unei mulțimi de programe de calculator (inițial generate aleator) care sunt modificate până în momentul în care este satisfăcută o condiție de oprire a procesului.

Terminologie folosită

- *Individ* (*cromozom*) - *program de calculator*. Respectând modelul inspirat de natură, elementele fundamentale ale unui individ (program) sunt *genele*. O înlănțuire de gene reprezintă un cromozom. Individul mai este numit, uneori, *soluție potențială* sau, pe scurt, *soluție*. Aceasta nu înseamnă că acel individ este chiar soluția problemei, ci înseamnă că este posibil ca, la sfârșitul rulării întregului algoritm de programare genetică, acel individ să conțină soluția corectă la problema respectivă.
- *Populație* - *mulțimea tuturor indivizilor considerați la un moment dat*. Din punct de vedere informatic, o populație nu este altceva decât un vector de indivizi (programe de calculator în cazul programării genetice).

Reprezentarea soluțiilor

Primul pas în rezolvarea unei probleme folosind programarea genetică este alegerea unei reprezentări eficiente pentru soluții.

Această alegere este dictată în majoritatea cazurilor de către problema care urmează a fi rezolvată.

Practic întrebarea este:

Cum putem reprezenta cât mai simplu și mai eficient un program de calculator?

Există foarte multe modalități de reprezentare. Cea mai populară modalitate de reprezentare este sub formă arborescentă.

Pentru a înțelege mai bine cum se pot reprezenta soluțiile (programele) în acest fel să considerăm mai întâi un exemplu.

Fie secvența de cod *C*:

```
int exemplu (int a)
{
  int temp1;
  int temp2;
  if (a > 10)
  {
    temp1 = 0;
  }
```

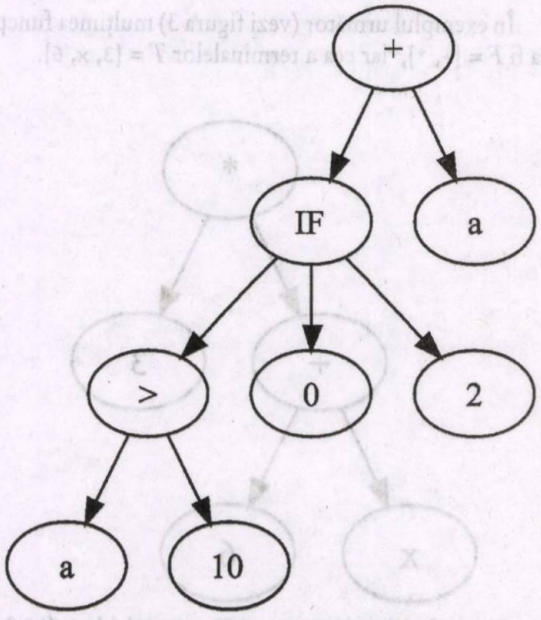


Figura 2: Reprezentarea arborescentă a unui program de calculator

```
  else
  {
    temp1 = 2;
  }
  temp2 = temp1 + a;
  return temp2;
}
```

Reprezentarea arborescentă care corespunde acestei secvențe este redată în figura 2.

După cum se poate observa, indivizii sunt reprezentați sub forma unei structuri arborescente care conține două tipuri de gene: ***funcții*** (***neterminale***) și ***terminale***.

Terminalele vor fi reprezentate de frunze (nodurile terminale), în timp ce funcțiile vor fi nodurile interne ale arborelui (nodurile care au unul sau mai mulți fii).

Numărul subarborilor unui nod care conține o funcție este dat de numărul argumentelor funcției respective.

Mulțimea funcțiilor (neterminalelor) poate include:

- *operatori aritmetici* (+, -, *, /)
- *funcții matematice* (`sin`, `cos`, `max`, `min`, etc.)
- *operatori condiționali* (`if - then - else`)
- *funcții pentru iterații* (`for`, `while`, `repeat`)
- *operatori logici* (`AND`, `OR`, `NOT`)
- alți operatori care pot fi utili problemei care trebuie rezolvată.

În exemplele următoare, pentru simplitatea reprezentării, nu vom mai folosi decât funcții aritmetice, celelalte fiind omise.

Mulțimea terminalelor conține variabile sau constante din program.

În exemplul următor (vezi figura 3) mulțimea funcțiior va fi $F = \{+, *\}$, iar cea a terminalelor $T = \{3, x, 6\}$.

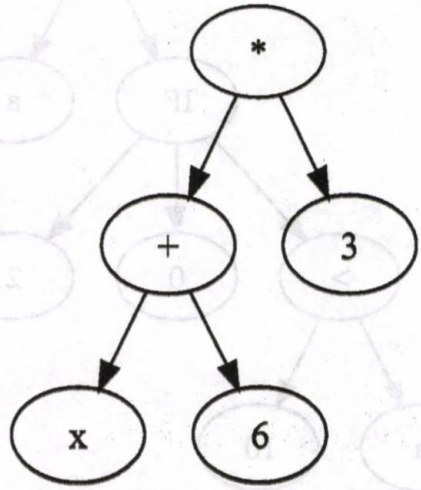


Figura 3: Arborele corespunzător expresiei (x + 6) * 3

Funcția * are două argumente: 3 și rezultatul returnat de valoarea subarborelui a carui radicină este +.

Funcția + are, de asemenea, două argumente: x și 6.

Expresia matematică ce corespunde acestui arbore este `3 * (x + 6)`.

Observație

Genele care vor fi folosite de algoritmul de programare genetică trebuie alese de către utilizator.

De obicei ele sunt date de intrările problemei la care se adaugă alte constante sau operatori care ar putea fi utili pentru rezolvarea problemei.

Funcția de calitate

În natură calitatea unui individ arată cât de bine este adaptat acel individ la mediul în care trăiește.

Cu cât are o calitate mai mare cu atât are șanse mai mari să supraviețuiască.

În cazul unui algoritm de programare genetică, această calitate a unui individ ne arată cât de bine rezolvă acel individ problema dată.

Pentru acest lucru avem nevoie de câteva instanțe ale problemei respective, împreună cu rezultatele așteptate. Acestea se numesc ***date de antrenament***.

Se aplică programul codificat într-un individ *PG* la problema de rezolvat pentru fiecare dintre datele de antrenament.

Se calculează diferența dintre ceea ce trebuie să se obțină și ceea ce s-a obținut în realitate. Această diferență va reprezenta calitatea unui program.

Funcția de calitate în această formă este o funcție de minimizare.

Aceasta înseamnă că se dorește obținerea unui program care să obțină rezultate cât mai apropiate de cele din mulțimea datelor de antrenament.

Exemplu:

Se consideră următoarea problemă: *Să se determine funcția f astfel încât:*

- *pentru* $x = 0$, $f(x) = 2$;
- *pentru* $x = 1$, $f(x) = 3$;
- *pentru* $x = 2$, $f(x) = 6$;
- *pentru* $x = 3$, $f(x) = 11$.

Presupunem că avem un individ care codifică expresia $E = x + 1$.

Dorim să determinăm cât de bună este calitatea acestui individ în raport cu datele de antrenament.

Introducând fiecare dintre intrări în expresia E obținem următoarele ieșiri:

- pentru $x = 0$, $E = 1$;
- pentru $x = 1$, $E = 2$;
- pentru $x = 2$, $E = 3$;
- pentru $x = 3$, $E = 4$.

Calitatea expresiei E se poate determina ca fiind suma diferențelor în valoare absolută dintre ceea ce s-a obținut și ceea ce se dorește a se obține.

Așadar, avem:

$$Calitate(E) = |1 - 2| + |2 - 3| + |3 - 6| + |4 - 11| = 12.$$

Pentru a fi o soluție a problemei date, expresia E ar trebui să aibă funcția de calitate egală cu 0 (avem de-a face cu o problemă de minimizare). Așadar, E nu rezolvă perfect această problemă.

Inițializare

După alegerea mulțimilor terminalelor și a funcțiilor primul pas al unui algoritm de programare genetică constă în crearea aleatoare a unei populații de programe. În cazul reprezentării arborescente, indivizii sunt generați pe baza unui algoritm recursiv care alege aleator, la fiecare pas, un terminal sau un neterminal.

Dacă este ales un terminal, algoritmul se oprește, iar dacă este aleasă o funcție, algoritmul va fi aplicat recursiv subarborilor corespuzători funcției din nod.

Din rațiuni de eficiență adâncimea arborilor generați este limitată la o valoare maximă.

Algoritmul de programare genetică nu este capabil să determine dintr-un singur pas soluția problemei. Din acest motiv, soluțiile generate inițial sunt modificate de-a lungul unui număr de iterații în speranța îmbunătățirii acestora.

Modificările aduse indivizilor se realizează cu ajutorul unor *operatori genetici* (*de variație*). Dintre acești operatori îi enumerăm doar pe cei mai importanți:

- *selecție*;
- *reproducere*;
- *recombinare*;
- *mutație*.

Selecție

Operația de ***selecție*** va consta într-o alegere a unui individ. În general, această alegere se realizează pe baza funcției de calitate, însă și o alegere aleatoare este permisă. Individul selectat este folosit în continuarea pentru aplicarea unuia din operatorii de reproducere, recombinare și mutație.

Cel mai popular mod de selecție este turnirul: se aleg doi indivizi aleator și cel mai bun dintre ei va fi returnat ca rezultat al selecției.

Reproducere

Reproducerea este operația prin care un individ este copiat (fără nicio modificare) în noua generație. În general 10% din populație participă la acest proces.

Recombinare

Operația de ***recombinare*** este cea care duce la diversificarea indivizilor dintr-o populație. Pentru recombinare se presupune că avem selectați doi indivizi (denumiți și *părinți*). Pașii pentru efectuarea acestei operații sunt următorii:

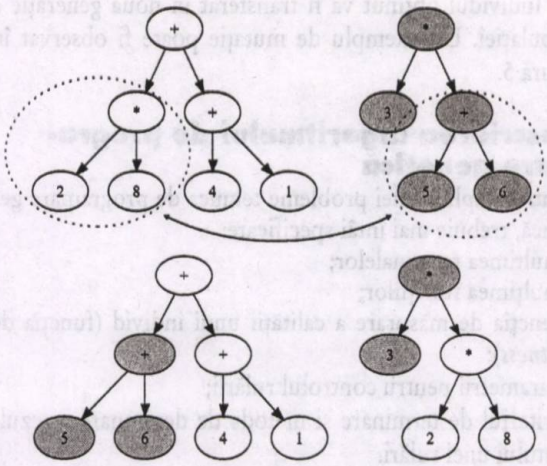


Figura 4: Operatorul de recombinare. Schimb de informație între doi indivizi

- se alege aleator un nod în primul individ;
- se alege aleator un nod în cel de a-l doilea individ;
- se interschimbă subarborii având ca rădăcini cele două noduri selectate anterior.

Indivizii rezultați în urma acestui proces vor fi transmiși în noua generație. Un exemplu de recombinare poate fi observat în figura 4.

Mutație

Operația de ***mutație*** introduce anumite modificări aleatoare în structura unui individ. În general modificările provocate prin mutație sunt mici pentru procesul evoluției, însă există cazuri când aceasta produce schimbări benefice.

Pașii pentru efectuarea acestei operații sunt următorii:

- se alege aleator un nod în individ;
- se șterge subarborele cu rădăcina în punctul considerat;
- se generează aleator un nou subarbore în locul subarborelui șters (acest lucru se face similar cu inițializarea indivizilor).

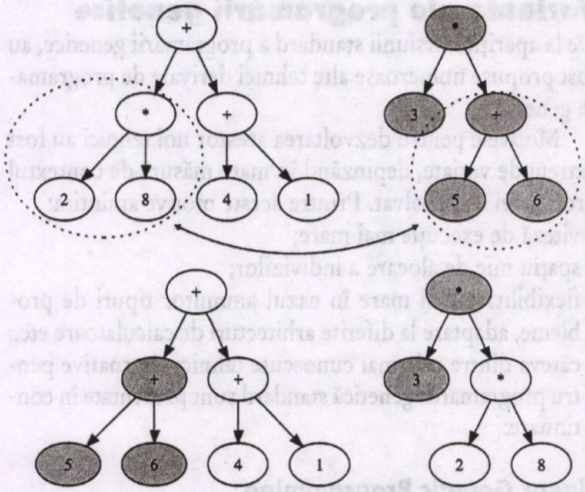


Figura 5: Operatorul de mutație. Un subarbore este șters, iar în locul lui este generat aleator un alt subarbore

Individul obținut va fi transferat în noua generație a populației. Un exemplu de mutație poate fi observat în figura 5.

Descrierea algoritmului de programare genetică

Pentru a aplica unei probleme tehnica de programare genetică, trebuie mai întâi specificate:

- mulțimea terminalelor;
- mulțimea funcțiilor;
- funcția de măsurare a calității unui individ (funcția de *fitness*);
- parametrii pentru controlul rulării;
- criteriul de terminare și metoda de desemnare a rezultatului unei rulări.

Acești pași sunt diferiți de la un caz la altul, în funcție de cerințele problemei, deci vor trebui specificați de fiecare dată de către utilizator. Aplicarea acestei tehnici se realizează după următorii pași:

- generarea populației inițiale de programe;
- repetarea succesivă a următorilor pași, până la întâlnirea criteriului de oprire:
  - ♦ rularea fiecărui program din populație și atribuirea unei valori de calitate raportată la cât de bine rezolvă un individ problema dată;
  - ♦ crearea unei noi populații de programe folosind operațiile de selecție, reproducere, recombinare și mutație; aceste operații sunt aplicate unui program ales cu o anumită probabilitate în funcție de calitatea sa; populația nou creată va conține:
    - ■ o copie a unor programe din generația anterioară;
    - ■ noile programe create prin aplicarea operatorilor genetici asupra programelor existente;
- cel mai bun program care apare la un moment dat (în oricare din generațiile de programe) va fi desemnat ca rezultat al algoritmului de programare genetică.

Variante ale programării genetice

De la apariția versiunii standard a programării genetice, au fost propuse numeroase alte tehnici derivate de programare genetică.

Motivele pentru dezvoltarea acestor noi tehnici au fost extrem de variate, depinzând în mare măsură de contextul problemei de rezolvat. Printre aceste motive amintim:

- viteză de execuție mai mare;
- spațiu mic de alocare a indivizilor;
- flexibilitate mai mare în cazul anumitor tipuri de probleme, adaptare la diferite arhitecturi de calculatoare etc.;
- câteva dintre cele mai cunoscute tehnici alternative pentru programarea genetică standard sunt prezentate în continuare:

Linear Genetic Programming

Linear Genetic Programming (*LGP*) - *programarea genetică liniară* - este o variantă a programării genetice care, după cum îi spune și numele, folosește o reprezentare liniară pentru indivizi.

LGP evoluează programe scrise într-un limbaj de programare imperativ (de exemplu *C*), nemaiținând cont de reprezentarea arborescentă folosită în cazul metodei *PG* standard.

Fiecare individ va fi o secvență de lungime variabilă de instrucțiuni simple dintr-un limbaj de programare.

Aceste instrucțiuni vor opera asupra unor variabile indexate (regiștri) sau asupra unor constante dintr-o mulțime predefinită.

Rezultatul va fi atribuit registrului destinație.

Exemplu

Următoarea secvență de cod ilustrează modul de reprezentare al unui program folosind *LGP*.

```
void LGP_program (double v[11])
{
  ...
  v[4] = v[2] - v[0];
  v[10] = v[1] / v[4];
  v[3] = sin(v[1]);
  v[7] = v[10] * v[3];
  v[9] = v[0] + v[7];
  ...
}
```

Dacă se consideră că registrii `v[0]`, `v[1]`, `v[2]` conțin valorile *a*, *b*, respectiv *c*, iar registrul `v[9]` conține rezultatul programului atunci cromozomul considerat va codifica expresia :

$$E = a + b / (c - a) \cdot \sin(b)$$

Multi Expression Programming

Multi Expression Programming (*MEP*) - programarea multi-expresie - este o variantă a programării genetice, care folosește o reprezentare liniară a cromozomului. Astfel, în acest caz indivizii vor fi formați din șiruri de gene ce pot codifica programe complexe de calculator.

Genele *MEP* sunt reprezentate de subșiruri de lungime variabilă. Numărul genelor dintr-un cromozom este constant, reprezentând lungimea cromozomului.

Fiecare genă poate codifica o funcție sau un simbol terminal.

O genă care conține o funcție va conține, de asemenea, și referințe către argumentul sau argumentele funcției respective.

Este demn de menționat faptul că argumentele unei funcții vor avea un indice cu o valoare mai mică decât a acesteia în cromozom.

Exemplu

Fie multimea de funcții $F = \{+, -, *\}$ și terminalele $T = \{a, b, c, d\}$.

Un exemplu de cromozom *MEP* este următorul:

```
1: a
2: b
3: + 1 2
4: c
5: d
6: - 4 5
7: * 3 6
```

Dacă parcurgem acest cromozom de sus în jos, atunci obținem următoarele expresii matematice:

- E_1 = a;
- E_2 = b;
- E_3 = a + b;
- E_4 = c;
- E_5 = d;
- E_6 = c - d;
- E_7 = (a + b) * (c - d).

Oricare dintre aceste expresii poate fi considerată o soluție a problemei.

În cazul *MEP* cea mai bună dintre aceste expresii va reprezenta acel individ.

Gene Expression Programming

Gene Expression Programming (*GEP*) - *programarea bazată pe mecanismul de exprimare a genelor* - este tot o variantă liniară a programării genetice.

În acest caz cromozomul va conține simboluri reprezentate de terminale și funcții.

Deși indivizii *GEP* sunt codificați liniar, cromozomii pot fi exprimați (transformați) într-o structură arborescentă, după cum se poate observa în figura 6.

Genele *GEP* sunt compuse dintr-un *cap* și o *coadă*. Scopul acestei separări este necesitatea generării de programe valide.

Capul conține atât terminale cât și funcții. Coada conține doar terminale.

Lungimea *h* a capului este aleasă de utilizator pentru fiecare problemă.

Lungimea cozii se calculează după formula $t = (n - 1) \cdot h + 1$, unde n este aritatea maximă a funcțiilor considerate.

Această lungime este astfel calculată încât toate funcțiile din capul unui cromozom să aibă valori atașate pentru toate argumentele.

Exemplu

Fie F = {+, -, *, /, sin}, T = {a, b, c} și expresia `a + b / (c - a) * sin(b)`.

Arborele corespunzător acestei expresii este dat în figura 6.

Dacă se parcurge în lățime acest arbore se va obține expresia necesară reprezentării *GEP*: `+a*/Sb-bca`.

Datorită faptului că doar în cap se pot folosi funcții, lungimea minimă a acestuia este 7.

Aritatea maximă a setului de funcții este 2, deci coada va avea lungimea minim 8.

Trebuie remarcat faptul că, în acest caz, pentru a reprezenta individul nu este nevoie de toată informația din coadă.

Un exemplu de cromozom *GEP* este:

+a*/Sb-bcaaabca,

unde partea evidențiată folosind caractere aldine reprezintă informația utilă.

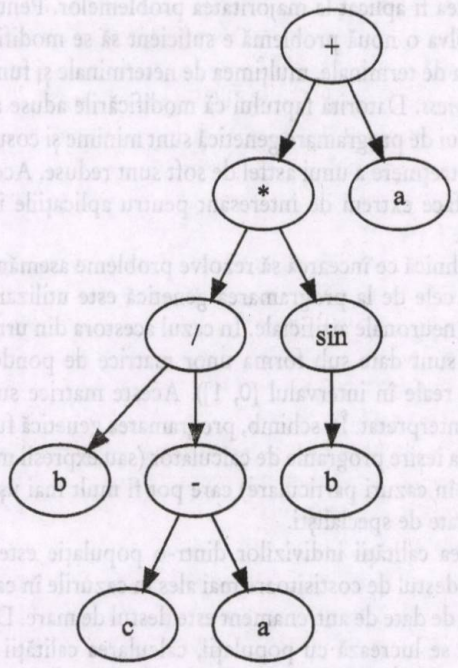


Figura 6: Arborele corespunzător expresiei a+b/(c-a)*sin(b)

Astfel, deși are o lungime fixă, nu toată informația din gena *GEP* este utilă.

Este posibil ca în unele cazuri să nu fie necesar să se folosească nici toată informația din cap. De exemplu, dacă pe prima poziție se află un terminal, atunci informația utilă se află doar în acest nod.

Aplicații ale programării genetice

Tehnica programării genetice a fost abordată în ultimii ani într-o mare gamă de aplicații.

În cele ce urmează va fi prezentată o listă cu cele mai importante domenii de aplicație:

- *procesarea documentelor și a imaginilor* (compresia imaginilor și a sunetelor, analiza imaginilor, recunoașterea limbajului natural, recunoașterea scrisului de mână etc.);
- *bioinformatică* (recunoașterea anumitor părți dintr-o proteină, detectarea automată a cancerului, clasificarea proteinelor nucleare etc.);
- *economie* (modelarea stocurilor, modelare financiară, clasificarea creditelor, anticiparea falimentului pe baza unor date financiare reale etc.);
- *circuite digitale și robotică* (proiectarea și controlul roboților, simularea locomoției bipede etc.);

- altele (evoluarea unor strategii de joc, rezolvarea identităților trigonometrice, rezolvarea ecuațiilor diferențiale, generarea automată de muzică etc.).

Puncte tari și puncte slabe ale programării genetice

Folosirea programării genetice pentru generarea codurilor sursă are o serie de avantaje importante:

- Algoritmul principal este unul suficient de general pentru a putea fi aplicat la majoritatea problemelor. Pentru a se rezolva o nouă problemă e suficient să se modifice mulțimea de terminale, mulțimea de neterminale și funcția de *fitness*. Datorită faptului că modificările aduse algoritmului de programare genetică sunt minime și costurile de întreținere a unui astfel de soft sunt reduse. Acest lucru îl face extrem de interesant pentru aplicațiile industriale.
- O altă tehnică ce încearcă să rezolve probleme asemănătoare cu cele de la programarea genetică este utilizarea rețelelor neuronale artificiale. În cazul acestora din urmă soluțiile sunt date sub forma unor matrice de ponderi (numere reale în intervalul [0, 1]). Aceste matrice sunt greu de interpretat. În schimb, programarea genetică furnizează la ieșire programe de calculator (sau expresii matematice în cazuri particulare) care pot fi mult mai ușor interpretate de specialiști.
- Calcularea calității indivizilor dintr-o populație este o operație destul de costisitoare mai ales în cazurile în care numărul de date de antrenament este destul de mare. Din cauză că se lucrează cu populații, calcularea calității se poate face în paralel, fiecare individ putând fi rulat pe un procesor. În acest fel timpul de rulare a algoritmului de PG poate fi redus simțitor.
- Tot din cauză că se lucrează cu populații la final se pot obține mai multe soluții posibile pentru problema de rezolvat. În acest fel utilizatorul va avea posibilitatea să aleagă acel program care îi este cel mai util.

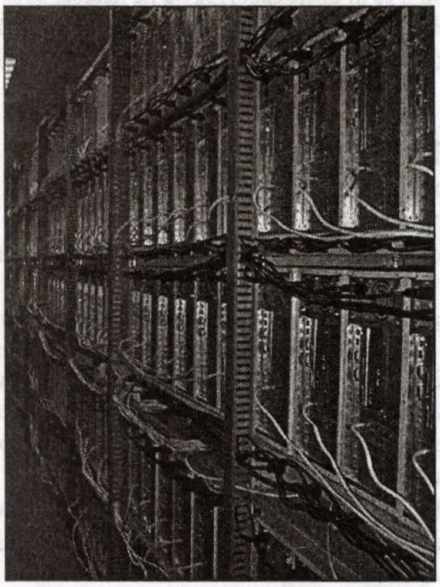


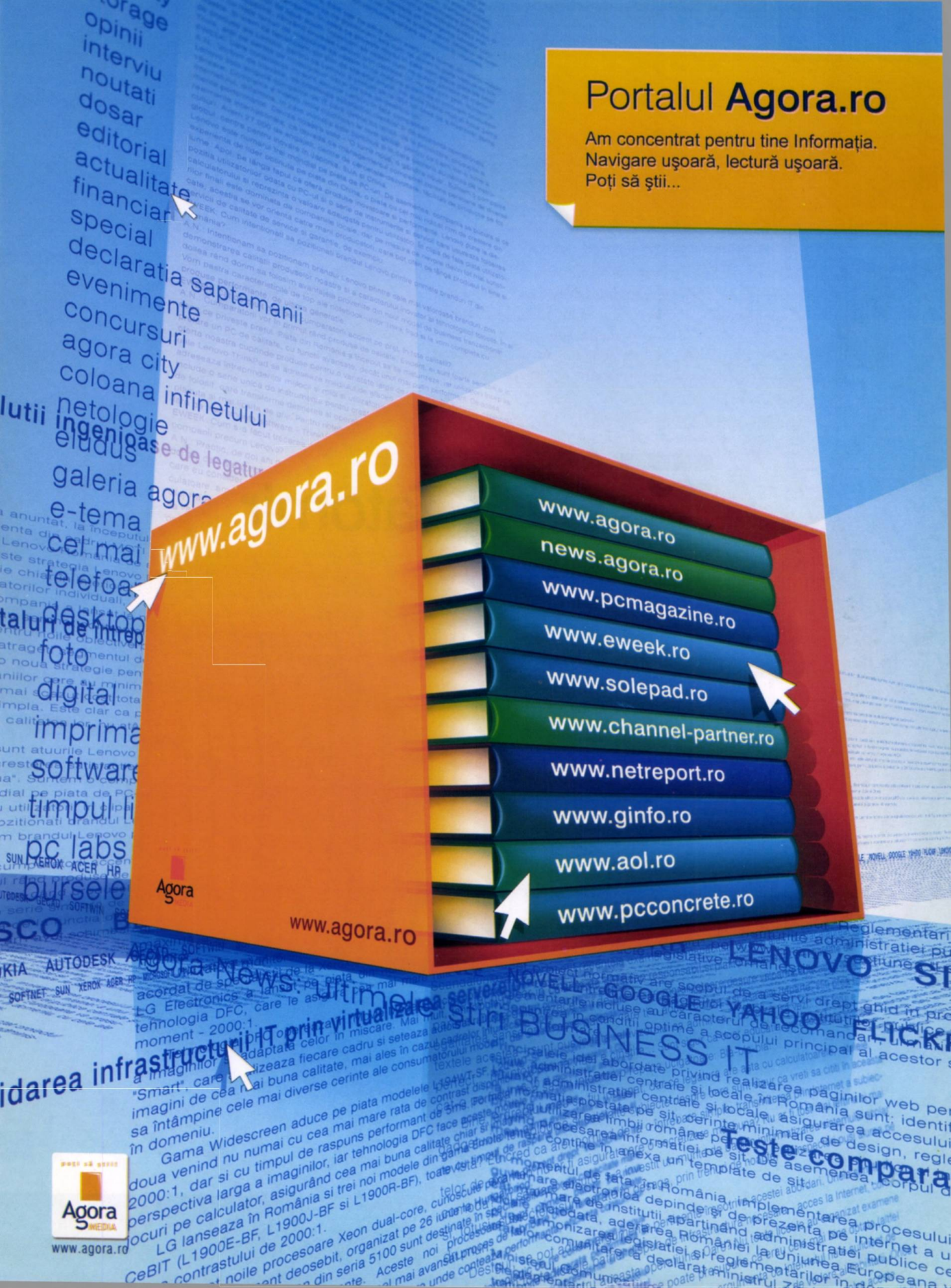
Figura 7: Rețeaua de calculatoare a lui John Koza pe care sunt rulați algoritmi de programare genetică

Pe lângă avantaje, programarea genetică are și o serie de dezavantaje:

- Nu se poate aplica pentru probleme foarte mari. De exemplu, se poate obține destul de simplu un program care sortează cinci numere, dar este imposibil să obținem un program brut care sortează 100 de numere. Acesta este doar un exemplu simplu, dar putem să ne imaginăm și alte aplicații mai complexe. De exemplu generarea automată a unui sistem de operare. Această sarcină este imposibil de realizat cu ajutorul programării genetice din cauză că spațiul de căutare (mulțimea tuturor codurilor sursă) este imensă.
- Tot din cauză că spațiul tuturor programelor posibile este prea mare, resursele necesare pentru a explora acest spațiu sunt destul de mari. În Figura 7 avem rețeaua de 1000 de calculatoare a lui *John Koza* pe care acesta rulează algoritmi de programare genetică. Pentru unele probleme, chiar și pe această rețea, timpul de execuție este de ordinul săptămânilor. Aceasta nu este însă o regulă generală, pentru că majoritatea cercetătorilor rulează programe de programare genetică pe simple calculatoare personale. Totul depinde de dificultatea problemei.
- Nu există baze teoretice pentru a garanta convergența. Asta înseamnă că există cazuri în care programarea genetică nu va găsi soluție pentru problema de rezolvat.
- Parametrii programului (dimensiunea populației, numărul de iterații) nu pot fi cunoscute de la început. Este recomandată rularea de mai multe ori a unui algoritm de programare genetică și ajustarea acestor parametri în funcție de rezultate.

Bibliografie

1. **M. Brameier, W. Banzhaf,** *A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining*, IEEE-TEVC, vol. 5, no. 1, pp. 17-26, 2001;
2. **C. Darwin,** *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, London: John Murray 1st edition, 1859;
3. **C. Ferreira,** *Gene Expression Programming: a New Adaptive Algorithm for Solving Problems*, Complex Systems, vol. 13, no. 2, pp. 87- 129, 2001;
4. **J. R. Koza,** *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992;
5. **J. R. Koza,** *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.
6. **M. Oltean, C. Groşan,** *Evolving Evolutionary Algorithms Using Multi Expression Programming*, in Proceedings of European Conference on Artificial Life: Advances in Artificial Life, ser. Lecture Notes in Artificial Intelligence, vol. 2801. Springer, pp. 651-658, 2003.


Portalul Agora.ro
Am concentrat pentru tine Informația.
Navigare uşoară, lectură uşoară.
Poți să ştii...
www.agora.ro
www.agora.ro
news.agora.ro
www.pcmagazine.ro
www.eweek.ro
www.solepad.ro
www.channel-partner.ro
www.netreport.ro
www.ginfo.ro
www.aol.ro
www.pcconcrete.ro
www.agora.ro

Cabinetul
Călin Tatomir

Noi ştim cum